Facultad de Ciencias

Lógica descriptiva $\mathcal{ALC}$: ontologías e inferencia.

(Description logic $\mathcal{ALC}$: ontologies and inference.)

Trabajo de Fin de Grado
para acceder al

Grado en Matemáticas

Autor: Adrián Pérez Herrero

Director: Inés González Rodríguez

Septiembre - 2019

Agradecimientos

Gracias a mi familia.
A mi madre,
por mostrarme el mundo
y enseñarme a apreciarlo.
A mi padre,
por darme ganas de entenderlo
y enseñarme que nada es serio.
Gracias,
por confiar en mis decisiones,
por el apoyo incondicional,
por los veranos en Liencres.
A Abram por compartir tanto y a María por transitividad.
A Ángela por su música y por crecer tan rápido.
A Aitor por el calor, los dibujos y los abrazos.
A Aroa por su energía, sus ganas y su felicidad.
Gracias a Julia y su familia,
por ser hogar y calor durante estos cuatro años,
por seguir navegando.
Gracias a Víctor,
por no rendirse nunca, por su confianza,
porque la carga pesa menos si se lleva entre dos.
Gracias a Eloy, Dani, Pablo, Ton,
David, Álvaro, Pedro, Diego,
por todas y cada una de las mañanas en clase,
por todas y cada una de las cenas hasta tarde,
por los badminton y las cervezas,
por hacer de una carrera un paseo.
Gracias a Inés, por hacer posible este trabajo,
por ayudarme a elegir dirección.
A compañeros, compañeras y profesores.
A Kings in the North+Bubrcia y Secta,
por una de las mejores experiencias de mi vida.
A Eila, Javieres, Sergio, Pachi, Andrea,
a VDR, HDLD y el Sonorama,
por estar ahí al volver.
A todos los que siguen a mi lado
y a los que ya no están.
A Aranda, Santander, Ljubljana,
Granada, Santiago
y lo que está por venir.
A mí, por disfrutar y aprender de ello.
Gracias.

"Peripeteia"

Resumen

Las lógicas descriptivas son una familia de lenguajes utilizados en Inteligencia Artificial para representar y explotar el conocimiento disponible sobre un dominio. En este trabajo abordamos la lógica descriptiva conocida como $\mathcal{ALC}$. Tras definir su sintaxis y semántica describimos cómo pueden utilizarse para representar conocimiento mediante ontologías. Planteamos diversos problemas de razonamiento a partir de ontologías y probamos que todos pueden reducirse a decidir sobre la consistencia de una ontología. Para resolver este problema de decisión introducimos un método basado en tableaux semánticos y demostramos que es correcto y completo.

Palabras clave

Inteligencia artificial, representación del conocimiento, lógica descriptiva $\mathcal{ALC}$, tableaux semánticos.

Abstract

Description logics constitute a family of languages used in Artificial Intelligence to represent and exploit the available knowledge of a domain of application. In this work we address one description logic known as $\mathcal{ALC}$. After defining its syntax and semantics, we show how they can be used to describe knowledge using ontologies. We introduce several reasoning problems from ontologies and we prove that they can all be reduced to an ontology-consistency problem. We explain a method to solve this problem based on semantic tableaux and we show that it is correct and complete.

Keywords

Artificial intelligence, kwnoledge representation, description logic $\mathcal{ALC}$, semantic tableaux.

Índice

Capítulo 1

Introducción

1.1. Motivación del trabajo

Las lógicas descriptivas (LDs) son una familia de lenguajes de representación del conocimiento usados para representar un dominio de una forma estructurada e inteligible [8]. El nombre *lógicas descriptivas* está motivado, por un lado, porque el dominio se representa mediante una base de conocimiento formada por descripciones compuestas de conceptos y relaciones entre conceptos; y por otro lado, porque estos lenguajes están dotados de una semántica basada en lógica.

Que la semántica esté basada en lógica significa que tenemos un procedimiento bien definido para saber cuándo un hecho se deduce de nuestra base de conocimiento. Además, gracias a ello, podemos usar algoritmos de razonamiento o inferencia para extraer información de la misma, como por ejemplo la técnica de *tableaux*, introducida por Schmidt-Schauß y Smolka [9]. Los conceptos a representar en nuestro dominio marcan la expresividad que debe alcanzar nuestra LD. Esta expresividad se modifica añadiendo constructores a nuestra sintaxis, lo que facilita, además, identificar fácilmente la semántica de las descripciones. Existe una relación entre expresividad y el coste computacional del razonamiento, cuanto más expresiva es una LD, más costoso resulta extraer información de la misma. Es por ello que se intenta establecer un equilibrio entre expresividad y complejidad. Esta expresividad suele estar restringida hasta el punto de no perder la decibilidad y la tratabilidad, al menos, en los razonamientos básicos.

La teoría de modelos estudia qué estructuras o dominios pueden describirse en función de los constructores que se empleen en una LD y las propiedades que se obtienen o se pierden según el conjunto de constructores utilizado. Además, estudiando estas propiedades se identifican cotas para la complejidad de las tareas de razonamiento.

El estudio de este campo creció desde la investigación en sistemas de representación del conocimiento junto con un intento de formalizar su semántica y los procedimientos de inferencia. Las LDs son la base de varios lenguajes de ontologías entre los que cabe destacar OWL, un lenguaje específico para ingeniería de ontologías que ha sido usado en campos tan ajenos como la agricultura (proyecto AGROVOC [10]), astronomía (proyecto en proceso de clasificación de cuerpos celestes [6]), educación (proyecto de planificación individualizada para alumnos con necesidades especiales [5]) y un largo etcétera que se puede consultar en la Sección 1.2 de [3] y en la sección 3.7 de [1].

1.2. Objetivos del trabajo

El objetivo general del trabajo es proporcionar una introducción al mundo de las lógicas descriptivas. Para ello, se fijan varios subobjetivos:

- Definir formalmente la sintaxis y semántica de una LD sencilla, $\mathcal{ALC}$.
- Estudiar cómo puede describirse un dominio con $\mathcal{ALC}$ a través de una base de conocimiento u ontología y qué problemas de inferencia o razonamiento pueden plantearse.
- Dar un algoritmo de inferencia y estudiar sus propiedades formales, en particular, corrección y completud.

1.3. Estructura del trabajo

En el Capítulo 2 se presenta una lógica descriptiva básica, la LD $\mathcal{ALC}$ *(Attributive concept Language with Complements)*, definiendo formalmente su sintaxis y semántica. En el Capítulo 3 se definen los conceptos de TBox y ABox, las partes terminológica y asertiva que conformarán una base de conocimiento. Además, se presentan las tareas de razonamiento en $\mathcal{ALC}$, que se identifica con un fragmento decidible de la lógica de primer orden. Por último, se verán algunas posibles ampliaciones de $\mathcal{ALC}$. En el Capítulo 4 se introducen algunas nociones y conceptos sobre teoría de modelos que nos serán necesarios para desarrollar el Capítulo 5 donde se presenta un algoritmo de tableaux para decidir sobre la consistencia de una base de conocimiento y se demuestra que es correcto, completo y termina. Finalmente, en el Capítulo 6 se presentan algunas conclusiones.

A lo largo de todo este trabajo se ha utilizado como referencia básica el libro de Baader et al. *An introduction to description logics* [3]. Otras referencias adicionales se indicarán en su contexto.

Capítulo 2

$\mathcal{ALC}$: una lógica descriptiva simple

La lógica descriptiva en la que nos vamos a centrar es $\mathcal{ALC}$, *Attributive concept Language with Complements*[1]. Se trata de una LD básica pero suficientemente expresiva que nos permitirá conocer el proceso de definir y trabajar con un lenguaje para la representación del conocimiento.

Al ser un modelo que queremos usar para describir un dominio (dígase situaciones de interés), debemos exponer en primer lugar los *elementos* que vamos a usar y más en concreto, los tipos de elementos que podemos tener y la forma de construirlos. Para ilustrar las definiciones y resultados y facilitar su comprensión usaremos un ejemplo basado en el dominio de un restaurante, expuesto en el esquema de la Figura 2.1. Para comenzar, encontraremos una diferenciación clara entre comida y personas. Estos dos conceptos principales se subdividirán en otros dependiendo de las características que queramos identificar. Sin embargo, no son conceptos independientes, sino que estarán relacionados; por ejemplo, una persona consume o cocina comida.

2.1. Sintaxis: conceptos, roles y descripciones

Para describir los elementos de un dominio y cómo se relacionan usaremos:

- **Conceptos** que representan conjuntos de elementos y pueden ser identificados con predicados unarios. Estos conceptos se construyen usando *nombres de conceptos* [2] y *nombres de roles* usando los constructores específicos de la lógica descriptiva con la que estemos trabajando. Llamaremos *extensión* de un concepto al conjunto de elementos que representa. Por ejemplo, *Arguiñano* está en la extensión de Cocinero.

- **Nombres de roles** que permiten una relación binaria entre elementos y que pueden ser identificados con predicados binarios. Diremos que b es un *r-valor* de a si el rol r relaciona a con b. Por ejemplo, si Repostero *especialista-en* Postre, decimos que Postre es un *especialista-en-valor* de Repostero.

[1] Nombre acuñado por Schmidt-Schauß y Smolka en [9], donde además se propuso el primer algoritmo basado en tableaux, que más tarde se amplió a otras LDs más complejas.

[2] Del inglés *concept names* y *role names*

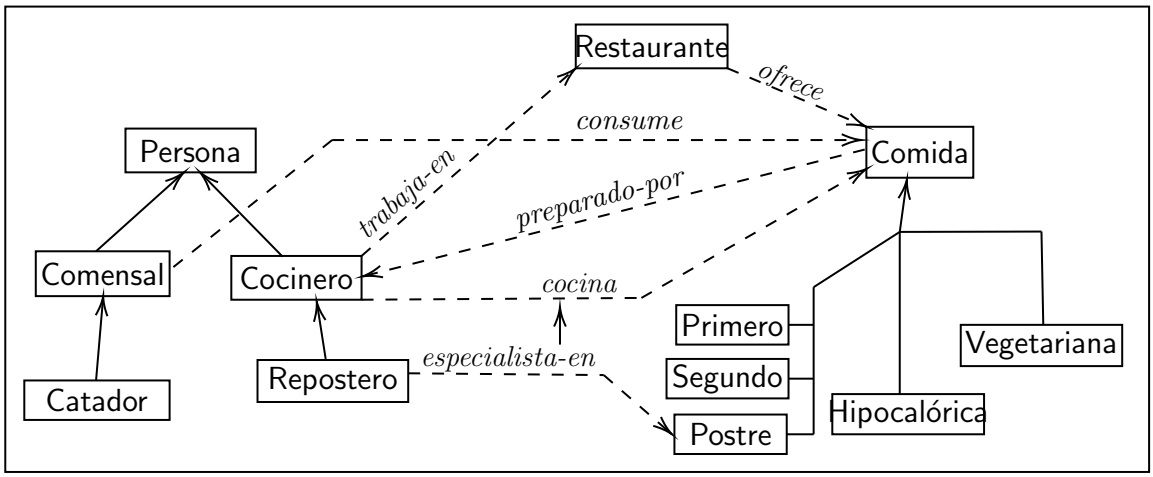

Figura 2.1: Representación del ejemplo.

Partiendo de estos elementos básicos y mediante la aplicación de diversos operadores se construye cualquier lógica descriptiva. En particular, la sintaxis de $\mathcal{ALC}$ se define como sigue:

Definición 2.1. *Sea* $\mathbf{C}$ *un conjunto de nombres de conceptos y* $\mathbf{R}$ *un conjunto de nombres de roles disjunto de* $\mathbf{C}$*. El* conjunto de conceptos $\mathcal{ALC}$ *sobre* $\mathbf{C}$ *y* $\mathbf{R}$ *está definido como sigue:*

I) *Todo nombre de concepto es un concepto* $\mathcal{ALC}$*.*

II) $\top$ *y* $\bot$ *son conceptos* $\mathcal{ALC}$*.*

III) *Si* C *y* D *son conceptos* $\mathcal{ALC}$ *y* r *es un nombre de rol, entonces los siguientes son también conceptos* $\mathcal{ALC}$*:*

$$\begin{array}{cl} C \sqcap D & \textit{(Conjunción)} \\ C \sqcup D & \textit{(Disyunción)} \\ \neg C & \textit{(Negación)} \\ \exists r.C & \textit{(Cuantificador existencial o restricción de existencia)} \\ \forall r.C & \textit{(Cuantificador universal o restricción de valor)} \end{array}$$

Se podrán usar paréntesis para esclarecer o dar prioridad.

Esta definición fija la sintaxis de nuestro modelo, así sabremos cuándo un concepto está bien formado o no. A partir de ahora se distinguirá entre conceptos *atómicos* o *compuestos*, los primeros serán los nombres de concepto y los conceptos del total $\top$ y el vacío $\bot$ y los segundos el resto de conceptos bien formados. Por defecto, se tenderá a utilizar A,B como conceptos atómicos, C,D como compuestos y r,s para roles.

Los conceptos $\mathcal{ALC}$ nos permiten modelar variedad de situaciones. Exponemos informalmente unos ejemplos de uso relacionados con el dominio de nuestro ejemplo para intentar esclarecer lo que se puede representar con estas cadenas de símbolos:

- La negación escrita como $\neg C$ puede ser leída como "no C" y describe todos los elementos que no están en la extensión de C. Por ejemplo $\neg$Cocinero incluye todo lo que no es un cocinero.

- La conjunción escrita como $C \sqcap D$ puede ser leída como "C y D" y describe los elementos que están en la extensión de C y en la extensión de D. Por ejemplo Cocinero$\sqcap$Comensal describe a aquellos elementos que son simultáneamente Cocinero y Comensal.

- La disyunción escrita como $C \sqcup D$ y puede ser leída como "C o D" y describe los elementos que están en la extensión de C, en la de D o en ambas. Como por ejemplo Primero $\sqcup$ Segundo corresponde a elementos que se consideren Primero o Segundo.

- El cuantificador universal, o restricción de valor escrita como $\forall r.C$ puede ser leída como "elementos tales que todos sus *r-valores* son C" y describe a todos los elementos que solo se relacionan mediante r con elementos de C. Como ejemplo, $\forall cocina.$Tarta se refiere a aquellos que solo cocinan Tarta.

- El cuantificador existencial o restricción de existencia escrita como $\exists r.C$ puede ser leída como "elementos tales que existe un *r-valor* que es un C" y describe a todos los elementos que están relacionados mediante r con al menos un elemento de C. Por ejemplo, $\exists cocina.$Tarta se refiere a aquellos que cocinan al menos una Tarta.

2.2. Semántica: la interpretación.

Una vez fijada la sintaxis para formar conceptos, necesitamos otorgar un significado a dichos concepto. De ello se ocupa la *semántica*. Para definir la semántica de manera formal vamos a usar lo que se denominará una *interpretación*. Una interpretación es una estructura que debe cumplir lo siguiente:

1. Contiene un conjunto no vacío al que llamaremos *dominio de la interpretación* cuyos elementos en ocasiones se denominan individuos u objetos.

2. Para cada *concepto simple* fija su extensión, es decir, si un elemento pertenece al concepto o no.

3. Para cada *rol* fija su extensión, es decir, qué par de elementos se encuentran relacionados por el rol.

Con todo ello procede la siguiente definición:

Definición 2.2. *Sea* $\mathbf{C}$ *un conjunto de nombres de conceptos simples y* $\mathbf{R}$ *un conjunto de nombres de roles disjunto de* $\mathbf{C}$. *Una* interpretación $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consiste en un conjunto no vacío* $\Delta^{\mathcal{I}}$ *llamado el* dominio de la interpretación *y una aplicación* $\cdot^{\mathcal{I}}$ *que lleva:*

a) todo concepto simple $A \in \mathbf{C}$ *a un conjunto* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;

b) todo rol $r \in \mathbf{R}$ *a una relación binaria* $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

La aplicación $\cdot^{\mathcal{I}}$ *se extiende a* $\top$, $\bot$ *y a conceptos compuestos como sigue. Sean* C, D *conceptos compuestos, entonces:*

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : \textit{ existe un } e \in \Delta^{\mathcal{I}} \textit{ con } (d,e) \in r^{\mathcal{I}} \textit{ y } e \in C^{\mathcal{I}}\} \\
(\forall r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : \textit{ para todo } e \in \Delta^{\mathcal{I}}, \textit{ si } (d,e) \in r^{\mathcal{I}}, \textit{ entonces } e \in C^{\mathcal{I}}\}
\end{aligned}$$

Diremos que:

- $C^{\mathcal{I}}$ *es la* extensión *de* C *en* $\mathcal{I}$*,*

- $b \in \Delta^{\mathcal{I}}$ *es un* r-valor *de* a *en* $\mathcal{I}$ *si* $(a,b) \in r^{\mathcal{I}}$

Cabe recalcar que la interpretación no está más restringida de lo descrito en la definición. En concreto, no está restringida a ningún cardinal, el dominio debe ser no vacío pero puede tener cardinal infinito; al igual sucede con la extensión de un concepto y la extensión de un rol.

Como ejemplo podemos considerar la siguiente interpretación representada en forma de gráfico en la Figura 2.2:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \{juan, jose, \textit{la-tortilla}, \textit{la-ensalada}, \textit{el-huevo}\} \\
\mathsf{Cocinero}^{\mathcal{I}} &= \{juan, jose\} \\
\mathsf{Comida}^{\mathcal{I}} &= \{\textit{la-tortilla}, \textit{la-ensalada}, \textit{el-huevo}\} \\
\mathsf{Persona}^{\mathcal{I}} &= \{juan, jose\} \\
\mathsf{Primero}^{\mathcal{I}} &= \{\textit{la-ensalada}\} \\
\textit{cocina}^{\mathcal{I}} &= \{(juan, \textit{la-ensalada}), (juan, \textit{la-tortilla}), (juan, \textit{el-huevo}), (jose, \textit{la-tortilla})\} \\
\textit{contiene}^{\mathcal{I}} &= \{(\textit{la-tortilla}, \textit{el-huevo}), (\textit{el-huevo}, \textit{el-huevo})\}
\end{aligned}$$

Por definición, todos los elementos están en $\top$ ya que $\Delta^{\mathcal{I}} = \top$. Los elementos *juan* y *jose* están en la extensión de Persona y *el-huevo* es un *contiene-valor* de sí mismo. Si usamos los constructores expuestos en la Definición 2.1, tenemos por ejemplo que *jose* está en la extensión de $\mathsf{Cocinero} \sqcap (\neg \exists \textit{cocina}.\mathsf{Primero})$. Por otro lado, nótese que $\forall \textit{cocina}.\mathsf{Comida}$ (los que sólo cocinan comida) no se corresponde, como cabría esperar, con los cocineros *juan* y *jose*, sino que su extensión es el dominio completo, todos los elementos. Esto se debe a que *la-ensalada*, *la-tortilla* y *el-huevo* no tienen ningún *cocina-valor* y por lo tanto estos *cocina-valores* cumplen todas las condiciones que les impongamos. En general, si un elemento no tiene *r-valores* con r un rol, entonces está en la extensión de $\forall r.C$ para cualquier concepto C. Por el contrario, *juan* no está en la extensión de $\forall \textit{cocina}.\mathsf{Primero}$ porque *juan* tiene un *cocina-valor* (*la-tortilla*) que no es Primero y si queremos identificar únicamente a los cocineros a partir del rol *cocina* y el concepto Comida tenemos que $\{juan, jose\} = (\forall \textit{cocina}.\mathsf{Comida} \sqcap \exists \textit{cocina}.\top)^{\mathcal{I}}$, es decir, los cocineros son los que cocinan algo y sólo cocinan comida.

Veamos ahora algunas propiedades básicas de la semántica.

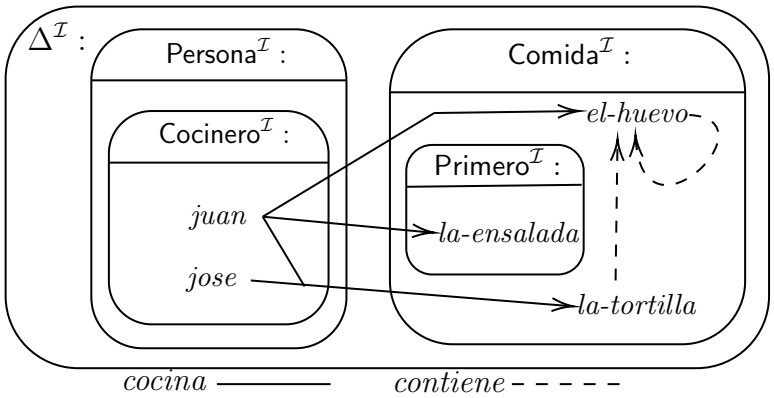

Figura 2.2: Gráfico de la interpretación.

Lema 2.3. *Sean $\mathcal{I}$ una interpretación, C y D conceptos y r un rol, entonces:*

$$\text{I)} \quad \top^{\mathcal{I}} = (C \sqcup \neg C)^{\mathcal{I}}$$

$$\text{II)} \quad \bot^{\mathcal{I}} = (C \sqcap \neg C)^{\mathcal{I}}$$

$$\text{III)} \quad \neg\neg C^{\mathcal{I}} = C^{\mathcal{I}}$$

$$\text{IV)} \quad \neg(C \sqcap D)^{\mathcal{I}} = (\neg C \sqcup \neg D)^{\mathcal{I}}$$

$$\text{V)} \quad \neg(C \sqcup D)^{\mathcal{I}} = (\neg C \sqcap \neg D)^{\mathcal{I}}$$

$$\text{VI)} \quad (\neg(\exists r.C))^{\mathcal{I}} = (\forall r.\neg C)^{\mathcal{I}}$$

$$\text{VII)} \quad (\neg(\forall r.C))^{\mathcal{I}} = (\exists r.\neg C)^{\mathcal{I}}$$

Demostración. Estas propiedades se derivan fácilmente de la Definición 2.2 y de propiedades básicas de conjuntos. En efecto:

I) Tenemos que $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ y por otro lado, $(C \sqcup \neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cup (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) = \Delta^{\mathcal{I}}$.

II) Tenemos que $\bot^{\mathcal{I}} = \emptyset$ y por otro lado, $(C \sqcap \neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}} = C^{\mathcal{I}} \cap (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) = \emptyset$.

III) Tenemos que $\neg\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (\neg C^{\mathcal{I}}) = \Delta^{\mathcal{I}} \setminus (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}})$ que es lo mismo que $C^{\mathcal{I}}$.

IV) Tenemos que $\neg(C \sqcap D)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (C \sqcap D)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (C^{\mathcal{I}} \cap D^{\mathcal{I}})$ y por las leyes de de Morgan, $\Delta^{\mathcal{I}} \setminus (C^{\mathcal{I}} \cap D^{\mathcal{I}}) = (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) \cup (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}) = (\neg C^{\mathcal{I}}) \cup (\neg D^{\mathcal{I}}) = (\neg C \sqcup \neg D)^{\mathcal{I}}$.

V) Tenemos que $\neg(C \sqcup D)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (C \sqcup D)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (C^{\mathcal{I}} \cup D^{\mathcal{I}})$ y por las leyes de Morgan $\Delta^{\mathcal{I}} \setminus (C^{\mathcal{I}} \cup D^{\mathcal{I}}) = (\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}) \cap (\Delta^{\mathcal{I}} \setminus D^{\mathcal{I}}) = (\neg C^{\mathcal{I}}) \cap (\neg D^{\mathcal{I}}) = (\neg C \sqcap \neg D)^{\mathcal{I}}$.

VI) Tenemos que $(\neg(\exists r.C))^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \{d \in \Delta^{\mathcal{I}} :$ existe un $e \in C^{\mathcal{I}}$ con $(d,e) \in r^{\mathcal{I}}\} = \{d \in \Delta^{\mathcal{I}} :$ no existe un $e \in C^{\mathcal{I}}$ con $(d,e) \in r^{\mathcal{I}}\} = \{d \in \Delta^{\mathcal{I}} :$ para todo $e \in \Delta^{\mathcal{I}}$,si $(d,e) \in r^{\mathcal{I}}$ entonces $e \notin C^{\mathcal{I}}\} = \{d \in \Delta^{\mathcal{I}} :$ para todo $e \in \Delta^{\mathcal{I}}$,si $(d,e) \in r^{\mathcal{I}}$ entonces $e \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\} = (\forall r.\neg C)^{\mathcal{I}}$.

VII) Tenemos que $(\neg(\forall r.C))^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash \{d \in \Delta^{\mathcal{I}} :$ para todo $e \in \Delta^{\mathcal{I}}$ si $(d,e) \in r^{\mathcal{I}}$ entonces $e \in C^{\mathcal{I}}\} = \{d \in \Delta^{\mathcal{I}} :$ existe un $e \in \Delta^{\mathcal{I}}$ con $(d,e) \in r^{\mathcal{I}}$ y $e \notin C^{\mathcal{I}}\} =$
$\{d \in \Delta^{\mathcal{I}} :$ existe un $e \in \Delta^{\mathcal{I}}$ con $(d,e) \in r^{\mathcal{I}}$ y $e \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\} = \{d \in \Delta^{\mathcal{I}} :$ existe un $e \in (\neg C)^{\mathcal{I}}$ con $(d,e) \in r^{\mathcal{I}}\} = (\exists r.\neg C)^{\mathcal{I}}$

□

Algunas de estas propiedades nos permiten reescribir conceptos , en algunos casos sin usar todos los constructores sin por ello sacrificar nuestra expresividad. Por ejemplo, es posible prescindir de la disyunción ya que, gracias a la propiedad V), cualquier concepto puede reescribirse usando en su lugar conjunción y negación sin cambiar su significado (es decir, su extensión). Esto también quiere decir que el conjunto de constructores dados en Definición 2.1 no es minimal. Se añaden, no obstante, estos constructores para facilitar la comprensión de las descripciones construidas, esto se suele denominar el *azúcar sintáctico*. Por otro lado, hay diversidad de combinaciones de constructores que dan lugar a lo que se denomina un *conjunto de constructores funcionalmente completo*, esto es, combinando estos constructores es posible generar todas las posibles extensiones de una expresión y por tanto cualquier descripción puede ser expresada en función de dichos constructores.

Capítulo 3

Ontologías en $\mathcal{ALC}$

El lenguaje $\mathcal{ALC}$ puede utilizarse para representar conocimiento sobre un dominio describiendo los distintos tipos de objetos que lo componen y cómo se relacionan entre sí. Esta resepresentación se denomina *base de conocimiento* u *ontología*. Las descripciones pueden realizarse, al menos, de cuatro formas diferentes:

a) Como si fuese un diccionario podemos definir un concepto a partir de otros, por ejemplo, podemos definir Cocinero y Repostero como:

$$\begin{aligned} \mathsf{Cocinero} &\equiv \mathsf{Persona} \sqcap \exists \mathit{cocina}.\mathsf{Comida} \\ \mathsf{Repostero} &\equiv \mathsf{Cocinero} \sqcap \forall \mathit{cocina}.\mathsf{Postre} \end{aligned}$$

es decir, un Cocinero es una persona que cocina al menos una comida y un Respostero, un cocinero que solo cocina postres.

b) Podemos introducir algunas restricciones sobre conceptos, por ejemplo, que un catador debe ser un comensal o que un primer plato no puede ser un postre:

$$\begin{aligned} \mathsf{Catador} &\sqsubseteq \mathsf{Comensal} \\ \mathsf{Primero} &\sqsubseteq \neg \mathsf{Postre} \end{aligned}$$

c) Podemos identificar con qué concepto (posiblemente compuesto) se corresponde un individuo concreto. Podemos por ejemplo afirmar que un individuo concreto que identificamos como "juan" es un cocinero que cocina primeros platos, es decir, es una instancia de $\mathsf{Cocinero} \sqcap \exists \mathit{cocina}.\mathsf{Primero}$.

d) Podemos relacionar individuos concretos mediante roles. Por ejemplo, podemos decir que *jose cocina la-tortilla*.

Así podemos distinguir dos componentes de una base de conocimiento: la que incluye descripciones genéricas, del tipo de a) y b), que denominaremos parte *terminológica* o **TBox**, y la que realiza afirmaciones sobre individuos concretos, del tipo c) y d), que denominaremos parte *asertiva* o **ABox**.

3.1. TBoxes en $\mathcal{ALC}$

Comenzamos la sección definiendo la sintaxis y la semántica de una **TBox**.

Definición 3.1. *Para C y D conceptos (simples o compuestos) una expresión del tipo $C \sqsubseteq D$ se denomina* inclusión de conceptos generales*, que abreviaremos GCI* [1]*. Usaremos también $C \equiv D$ como abreviatura de $C \sqsubseteq D$ y $D \sqsubseteq C$.*

Definimos una TBox *de $\mathcal{ALC}$ como una colección finita de GCIs.*

Diremos que una interpretación $\mathcal{I}$ satisface *una GCI $C \sqsubseteq D$ si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Una interpretación que satisface todas las GCI de una TBox $\mathcal{T}$ se denomina un* modelo *de $\mathcal{T}$.*

Denominaremos axioma *a cualquier expresión en una TBox del estilo $C \sqsubseteq D$ o $C \equiv D$, pudiendo referirnos a la primera como* axioma de inclusión *y a la segunda como* axioma de equivalencia.

La interpretación dada en el Ejemplo 2.2 satisface cada una de las GCIs de:

$$\begin{aligned}\mathcal{T}_1 = \{&\mathsf{Cocinero} \sqsubseteq \mathsf{Persona},\\ &\mathsf{Comida} \sqsubseteq \neg\mathsf{Persona},\\ &\mathsf{Cocinero} \sqsubseteq \exists \mathit{cocina}.\mathsf{Comida},\\ &\mathsf{Primero} \sqsubseteq \mathsf{Comida}\}\end{aligned}$$

y por lo tanto es un modelo de $\mathcal{T}_1$. Para comprobarlo para cada GCI $C \sqsubseteq D$, debemos determinar $C^{\mathcal{I}}$ y $D^{\mathcal{I}}$ y comprobar si realmente $C^{\mathcal{I}}$ es un subconjunto de $D^{\mathcal{I}}$. En nuestro caso, es fácil comprobar que $\mathsf{Cocinero}^{\mathcal{I}} = \{\mathit{jose}, \mathit{juan}\} \subseteq \{\mathit{jose}, \mathit{juan}\} = \mathsf{Persona}^{\mathcal{I}}$. Para la segunda GCI tenemos $\mathsf{Comida}^{\mathcal{I}} = \{\textit{el-huevo}, \textit{la-ensalada}, \textit{la-tortilla}\} \subseteq \{\textit{el-huevo}, \textit{la-ensalada}, \textit{la-tortilla}\} = (\neg\mathsf{Persona})^{\mathcal{I}}$. Para la tercera tenemos $\mathsf{Cocinero}^{\mathcal{I}} = \{\mathit{juan}, \mathit{jose}\} = (\exists \mathit{cocina}.\mathsf{Comida})^{\mathcal{I}}$ y para la cuarta es obvio que $\mathsf{Primero}^{\mathcal{I}} = \{\textit{la-ensalada}\} \subseteq \{\textit{el-huevo}, \textit{la-ensalada}, \textit{la-tortilla}\} = \mathsf{Comida}^{\mathcal{I}}$.

Por el contrario, si generamos una nueva TBox $\mathcal{T}_2$ como:

$$\mathcal{T}_2 = \mathcal{T}_1 \cup \{\mathsf{Comida} \sqsubseteq \exists \mathit{contiene}.\mathsf{Comida}\}$$

entonces tendremos que la interpretación del Ejemplo 2.2 no es un modelo de $\mathcal{T}_2$ ya que $\mathsf{Comida}^{\mathcal{I}} = \{\textit{el-huevo}, \textit{la-ensalada}, \textit{la-tortilla}\} \not\subseteq \{\textit{el-huevo}, \textit{la-tortilla}\} = (\exists \mathit{contiene}.\mathsf{Comida})^{\mathcal{I}}$, porque *la-ensalada* no está relacionada mediante *contiene* con ninguna otra comida en nuestro ejemplo.

En general, según vayamos incluyendo más GCIs en una TBox, vamos reduciendo el número de modelos de dicha TBox.

Lema 3.2. *Si $\mathcal{T} \subseteq \mathcal{T}'$ siendo $\mathcal{T}$ y $\mathcal{T}'$ dos TBoxes, entonces todo modelo de $\mathcal{T}'$ es también un modelo de $\mathcal{T}$.*

Demostración. Resulta inmediato ya que dado $\mathcal{I}$ un modelo de $\mathcal{T}'$ sabemos, por la Definición 3.1, que esta interpretación cumple todas las GCI en $\mathcal{T}'$ y, al estar cada GCI de $\mathcal{T}'$ en $\mathcal{T}$, tenemos que $\mathcal{I}$ también cumple todas las GCI en $\mathcal{T}$. Por tanto, $\mathcal{I}$ es un modelo de $\mathcal{T}$. □

[1] Del inglés *general concept inclusion.*

$$
\begin{aligned}
\mathcal{T}_e = \{&\mathsf{Cocinero} \sqsubseteq \mathsf{Persona} \sqcap \exists \mathit{cocina}.\mathsf{Comida}, && (\mathcal{T}_e.1)\\
&\exists \mathit{cocina}.\top \sqsubseteq \mathsf{Cocinero}, && (\mathcal{T}_e.2)\\
&\mathsf{Comensal} \sqsubseteq \mathsf{Persona} \sqcap \neg \mathsf{Cocinero}, && (\mathcal{T}_e.3)\\
&\mathsf{Catador} \sqsubseteq \mathsf{Comensal}, && (\mathcal{T}_e.4)\\
&\mathsf{Comida} \sqsubseteq \neg \mathsf{Persona}, && (\mathcal{T}_e.5)\\
&\mathsf{Primero} \sqsubseteq \mathsf{Comida}, && (\mathcal{T}_e.6)\\
&\mathsf{Segundo} \sqsubseteq \mathsf{Comida}, && (\mathcal{T}_e.7)\\
&\mathsf{Postre} \sqsubseteq \mathsf{Comida}, && (\mathcal{T}_e.8)\\
&\mathsf{Verdura} \sqsubseteq \mathsf{Comida} \sqcap \neg \mathsf{Carne}, && (\mathcal{T}_e.9)\\
&\mathsf{Repostero} \sqsubseteq \mathsf{Cocinero} \sqcap \exists \mathit{especialista\text{-}en}.\mathsf{Postre}\} && (\mathcal{T}_e.10)
\end{aligned}
$$

Figura 3.1: Ejemplo de TBox que modela el ejemplo de la Figura 2.1.

En la Figura 3.1 definimos una TBox $\mathcal{T}_e$ que intenta modelar nuestro ejemplo del restaurante presentado en la Figura 2.1. En dicha TBox queda reflejado conocimiento genérico sobre el dominio. En los axiomas $\mathcal{T}_e.1$ y $\mathcal{T}_e.2$, que un cocinero es una persona que cocina y que cualquiera que cocine ha de ser un cocinero. En el axioma $\mathcal{T}_e.3$, que un comensal ha de ser una persona que no sea cocinero; en el axioma $\mathcal{T}_e.4$, que un catador es un comensal; en el axioma $\mathcal{T}_e.5$, que la comida no es una persona; en los axiomas $\mathcal{T}_e.6$, $\mathcal{T}_e.7$ y $\mathcal{T}_e.8$, que el primer plato, el segundo plato y el postre son comida y en el axioma $\mathcal{T}_e.10$, que el repostero es un cocinero y es especialista en algún postre. El axioma $\mathcal{T}_e.9$ no se encuentra representado en el ejemplo de la Figura 2.1 pero indica que la verdura es un tipo de comida y que no es carne.

3.2. ABoxes en $\mathcal{ALC}$

Una TBox proporciona descripciones genéricas sobre un dominio que pueden complementarse con conocimiento específico sobre individuos concretos en lo que denominamos ABox.

Definición 3.3. *Sea* $\mathbf{I}$ *un conjunto de nombres de individuos disjunto de* $\mathbf{R}$ *y* $\mathbf{C}$*. Sean* $a, b \in \mathbf{I}$*, C un concepto (posiblemente compuesto) y* $r \in \mathbf{R}$ *un rol, entonces definimos un* aserto *como una expresión de una de las siguientes formas:*

- *a: C, que denominaremos un* aserto de conceptos $\mathcal{ALC}$*.*
- *(a,b): r, que denominaremos un* aserto de roles $\mathcal{ALC}$*.*

También se pueden denominar simplemente aserto de conceptos o aserto de roles.

Una ABox *en* $\mathcal{ALC}$ *es un conjunto finito de asertos.*

Una interpretación $\mathcal{I}$ *se extiende a* $\mathbf{I}$ *haciendo corresponder cada nombre de individuo a con un elemento* $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*. También diremos que:*

- $\mathcal{I}$ satisface un aserto de conceptos*, a: C si y solo si* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ *y,*

- $\mathcal{I}$ satisface un aserto de roles, *(a,b): r si y solo si* $(a,b)^{\mathcal{I}} \in r^{\mathcal{I}}$.

Una interpretación que satisface todos los asertos de una ABox $\mathcal{A}$ *se denomina* modelo *de* $\mathcal{A}$.

En la Figura 3.3 presentamos un ejemplo de ABox con asertos de conceptos y de roles modelando el ejemplo de la Figura 2.2. En dicha ABox se recoge conocimiento específico sobre individuos concretos como en los asertos $\mathcal{A}.1$, $\mathcal{A}.2$ y $\mathcal{A}.3$, que denotan que juan, jose y alberto son personas. En los asertos $\mathcal{A}.4$, $\mathcal{A}.5$ y $\mathcal{A}.6$ se refleja que la ensalada, el huevo y la tortilla son comida y que la ensalada en concreto es un primer plato. Por último en los asertos $\mathcal{A}.7$, $\mathcal{A}.8$, $\mathcal{A}.9$ y $\mathcal{A}.10$ se recogen las relaciones de que el huevo y la tortilla contienen huevo y que juan cocina la ensalada y jose la tortilla.

$$
\begin{aligned}
\mathcal{A} = \{&\mathit{juan} : \mathsf{Persona}, && (\mathcal{A}.1)\\
&\mathit{jose} : \mathsf{Persona}, && (\mathcal{A}.2)\\
&\mathit{alberto} : \mathsf{Persona}, && (\mathcal{A}.3)\\
&\mathit{la\text{-}ensalada} : \mathsf{Comida} \sqcap \mathsf{Primero}, && (\mathcal{A}.4)\\
&\mathit{el\text{-}huevo} : \mathsf{Comida}, && (\mathcal{A}.5)\\
&\mathit{la\text{-}tortilla} : \mathsf{Comida}, && (\mathcal{A}.6)\\
&(\mathit{el\text{-}huevo}, \mathit{el\text{-}huevo}) : \mathit{contiene}, && (\mathcal{A}.7)\\
&(\mathit{la\text{-}tortilla}, \mathit{el\text{-}huevo}) : \mathit{contiene}, && (\mathcal{A}.8)\\
&(\mathit{juan}, \mathit{la\text{-}ensalada}) : \mathit{cocina}, && (\mathcal{A}.9)\\
&(\mathit{jose}, \mathit{la\text{-}tortilla}) : \mathit{cocina}\} && (\mathcal{A}.10)
\end{aligned}
$$

Figura 3.2: $\mathcal{A}_1$: Una posible ABox para nuestro ejemplo representado en la Figura 2.1.

Es fácil comprobar que la siguiente interpretación $\mathcal{I}$ es un modelo de la ABox $\mathcal{A}_1$ de la Figura 3.2.

$$
\begin{aligned}
\Delta^{\mathcal{I}} &= \{j, a, en, h, to\}, & \mathsf{Cocinero}^{\mathcal{I}} &= \emptyset,\\
\mathit{jose}^{\mathcal{I}} &= \mathit{juan}^{\mathcal{I}} = j, & \mathsf{Comensal}^{\mathcal{I}} &= \{a\},\\
\mathit{alberto}^{\mathcal{I}} &= a, & \mathsf{Comida}^{\mathcal{I}} &= \{en, h, to\},\\
\mathit{la\text{-}ensalada}^{\mathcal{I}} &= en, & \mathsf{Primero}^{\mathcal{I}} &= \{en, to\},\\
\mathit{el\text{-}huevo}^{\mathcal{I}} &= h, & \mathsf{Segundo}^{\mathcal{I}} &= \{h\},\\
\mathit{la\text{-}tortilla}^{\mathcal{I}} &= to, & \mathit{cocina}^{\mathcal{I}} &= \{(j, en), (j, to), (j, h)\},\\
\mathsf{Persona}^{\mathcal{I}} &= \{j, a\}, & \mathit{contiene}^{\mathcal{I}} &= \{(ca, ca), (to, h)\}.
\end{aligned}
$$

Además, nótese que identificamos los individuos concretos *juan* y *jose* con el mismo elemento j. Según nuestra semántica, esto es válido; sin embargo, otras lógicas descriptivas implementan la llamada Hipótesis de Unicidad del Nombre [2], que requiere que si $a \neq b$, entonces $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. Por otro lado, podemos observar que en la interpretación hay más

[2]Del Inglés, UNA *Unique Name Assumption*

$$
\begin{aligned}
\mathcal{A}_e = \{&juan : \mathsf{Persona}, && (\mathcal{A}_e.1)\\
&jose : \mathsf{Persona} \sqcap \mathsf{Cocinero}, && (\mathcal{A}_e.2)\\
&alberto : \mathsf{Persona} \sqcap \mathsf{Comensal}, && (\mathcal{A}_e.3)\\
&la\text{-}ensalada : \mathsf{Comida} \sqcap \mathsf{Primero}, && (\mathcal{A}_e.4)\\
&la\text{-}carne : \mathsf{Comida}, && (\mathcal{A}_e.5)\\
&tortilla1 : \mathsf{Comida}, && (\mathcal{A}_e.6)\\
&tortilla2 : \mathsf{Comida}, && (\mathcal{A}_e.7)\\
&(el\text{-}huevo, el\text{-}huevo) : contiene, && (\mathcal{A}_e.8)\\
&(juan, la\text{-}ensalada) : cocina, && (\mathcal{A}_e.9)\\
&(juan, tortilla1) : cocina\} && (\mathcal{A}_e.10)
\end{aligned}
$$

Figura 3.3: $\mathcal{A}_e$: ABox alternativa para el ejemplo expuesto en Figura 2.1.

elementos en la extensión de Primero de los exigidos en la ABox, ya que el aserto $\mathcal{A}.4$ exige que *la-ensalada* sea un Primero, pero no se lo exige a *la-tortilla*. Además, interpreta el concepto de Comensal aunque no se menciona en la ABox. Estas son situaciones un poco forzadas pero también permitidas por nuestra definición de semántica. También cabe recalcar que esta interpretación no es un modelo de la TBox definida en la Figura 3.1 ya que, por ejemplo, $j \in \exists cocina.\top$ y sin embargo $j \notin \mathsf{Cocinero}$. Por lo tanto, viola el axioma $\mathcal{T}_e.2$ y no es un modelo de $\mathcal{T}_e$.

Análogamente a lo que ocurría con las TBoxes, añadir asertos reduce el conjunto de modelos de una ABox.

Lema 3.4. *Si $\mathcal{A} \subseteq \mathcal{A}'$ siendo $\mathcal{A}$ y $\mathcal{A}'$ dos ABoxes, entonces todo modelo de $\mathcal{A}'$ es también un modelo de $\mathcal{A}$.*

Demostración. Resulta inmediato ya que dado $\mathcal{I}$ un modelo de $\mathcal{A}'$ sabemos, por la Definición 3.3, que esta interpretación cumple todos los asertos en $\mathcal{A}'$ y, al estar cada aserto de $\mathcal{A}'$ en $\mathcal{A}$, tenemos que $\mathcal{I}$ también cumple todas los asertos en $\mathcal{A}$. Por tanto, $\mathcal{I}$ es un modelo de $\mathcal{A}$. □

3.3. Bases de conocimiento en $\mathcal{ALC}$

Una TBox y una ABox pueden combinarse para formar una base de conocimiento.

Definición 3.5. *Una* base de conocimiento *u* ontología $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *es un par formado por una ABox $\mathcal{A}$ y una TBox $\mathcal{T}$. Una interpretación que sea un modelo tanto de $\mathcal{A}$ como de $\mathcal{T}$ es un* modelo *de $\mathcal{K}$.*

Según esta definición, para que una interpretación sea un modelo de $\mathcal{K}$ debe satisfacer todos los asertos de la ABox $\mathcal{A}$ y todas las GCIs de la TBox $\mathcal{T}$. Por ejemplo, si tomamos $\mathcal{K}_e = (\mathcal{T}_e, \mathcal{A}_e)$ con $\mathcal{A}_e$ definida en la Figura 3.3, como hemos comprobado antes la interpretación $\mathcal{I}$ anterior no es un modelo de $\mathcal{T}_e$ y por lo tanto no es modelo de $\mathcal{K}_e$. Podemos, sin embargo, construir una interpretación $\mathcal{I}'$, representada en el diagrama de la Figura 3.4, que sea un modelo de la base de conocimiento $\mathcal{K}_e$ como sigue:

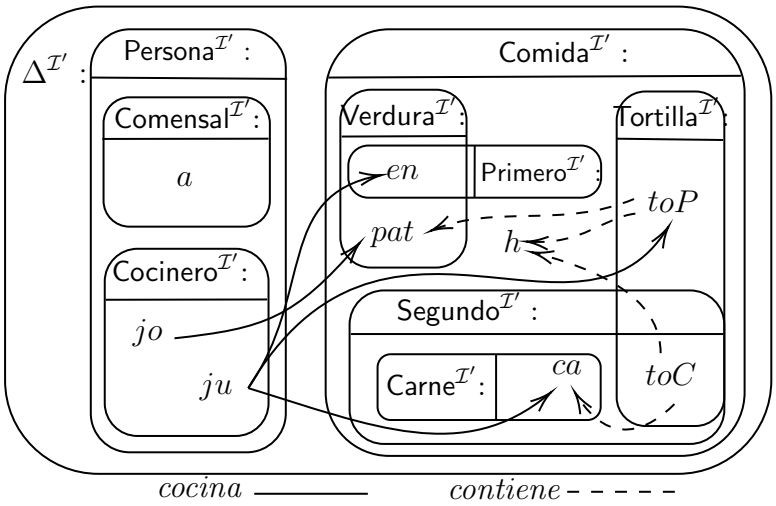

Figura 3.4: Diagrama de la interpretación $\mathcal{I}'$ exceptuando los nombres de individuos.

$$
\begin{aligned}
\Delta^{\mathcal{I}'} &= \{jo, ju, a, en, ca, \\
&\quad toC, toP, pat, h\}, \\
jose^{\mathcal{I}'} &= jo, \\
juan^{\mathcal{I}'} &= ju, \\
alberto^{\mathcal{I}'} &= a, \\
\textit{el-huevo}^{\mathcal{I}'} &= h, \\
\textit{la-carne}^{\mathcal{I}'} &= ca, \\
tortilla1^{\mathcal{I}'} &= toP, \\
tortilla2^{\mathcal{I}'} &= toC, \\
\textit{la-ensalada}^{\mathcal{I}'} &= en, \\
\mathsf{Verdura}^{\mathcal{I}'} &= \{en, pat\},
\end{aligned}
\qquad
\begin{aligned}
\mathsf{Carne}^{\mathcal{I}'} &= \{ca\}, \\
\mathsf{Tortilla}^{\mathcal{I}'} &= \{toP, toC\}, \\
\mathsf{Persona}^{\mathcal{I}'} &= \{ju, jo, a\}, \\
\mathsf{Cocinero}^{\mathcal{I}'} &= \{ju, jo\}, \\
\mathsf{Comensal}^{\mathcal{I}'} &= \{a\}, \\
\mathsf{Comida}^{\mathcal{I}'} &= \{en, ca, toC, toP\}, \\
\mathsf{Primero}^{\mathcal{I}'} &= \{en\}, \\
\mathsf{Segundo}^{\mathcal{I}'} &= \{ca, toC\}, \\
cocina^{\mathcal{I}'} &= \{(ju, en), (ju, toP), (ju, ca), (jo, pat)\}, \\
contiene^{\mathcal{I}'} &= \{(toC, ca), (toP, pat), (h, h)\}.
\end{aligned}
$$

Cabe notar que en nuestra ABox $\mathcal{A}_e$ de la Figura 3.3 hemos definido a *jose* como un cocinero en el aserto ($\mathcal{A}_e$.2), y entonces, por el axioma ($\mathcal{T}_e$.1) de la TBox sabemos que cocina al menos un plato de Comida. Nótese sin embargo que no tenemos ningún aserto del tipo $(jose, x) : cocina$, con $x \in \mathbf{I}$ que además debe cumplir también el aserto $x : \mathsf{Comida}$ aunque podamos deducir que su existencia es necesaria para tener un modelo de $\mathcal{T}_e$ (y de hecho en la interpretación $\mathcal{I}'$ tenemos $(jo, pat) \in cocina^{\mathcal{I}'}$). Con este ejemplo surge una diferencia esencial con respecto a las bases de datos. En una base de datos podemos imponer una *restricción de integridad* que obligue la existencia de un plato cocinado por cualquier elemento interpretado como Cocinero para conservar la integridad del modelo representado. Sin embargo, en nuestras bases de conocimiento, esta falta de información no provoca ninguna falta de integridad ya que no estamos considerando una sola ABox sino todos los modelos que se ajusten a nuestra base $\mathcal{K}_e$. En algunas de estas ABoxes podrá

figurar que *jose cocina* un plato en concreto y en otras sólo sabremos de la existencia de un plato cocinado por *jose* pero no tendremos representación del mismo en la ABox.

Estas diferencias se ven plasmadas también cuando tenemos en cuenta que los nombres de individuos pueden tener diferentes propiedades ajustándose al mismo modelo, dependiendo de la interpretación. Supongamos, por ejemplo, que añadimos el axioma a nuestra TBox:

$$\mathsf{Vegetariana} \equiv \mathsf{Comida} \sqcap \forall contiene.\neg\mathsf{Carne}$$

Siguiendo este axioma en nuestra interpretación tenemos que en el concepto Vegetariana está el elemento toP, interpretación del nombre de individuo *tortilla1* ya que todo lo que contiene es $\{pat, h\} \subseteq \neg\mathsf{Carne}^{\mathcal{I}'}$. Pero esta pertenencia es intrínseca a la interpretación, no se debe asumir que esta propiedad con respecto a *tortilla1* es compartida por todos los modelos, ya que podríamos tener un modelo en el cual añadimos algún ingrediente interpretado como Carne y nuestro elemento se saldría de este concepto.

El principio que contempla estos dos casos se conoce como *Hipótesis del mundo abierto*[3]. Se asume en las bases de conocimiento de LDs mientras que en paradigmas de representación del conocimiento clásicas (como en los sistemas basados en reglas de lógica proposicional o de predicados y en programación lógica) se suele asumir la *Hipótesis de mundo cerrado*[4]. Esto genera, por un lado, un aumento de la dificultad al razonar con nuestras bases de conocimiento y, a la vez un aumento de las situaciones contempladas, véase Sección 2.2.4.4 de [2].

3.4. Problemas básicos de razonamiento

Una vez definidas las bases de conocimiento en $\mathcal{ALC}$, nos disponemos a exponer los problemas de razonamiento más comunes que se suelen plantear sobre estas bases.

Definición 3.6. *Sean $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ una base de conocimiento, C, D conceptos y b un nombre de individuo. Entonces decimos que:*

I) *C es* satisfacible con respecto a $\mathcal{T}$ *si y solo si existen un modelo $\mathcal{I}$ de $\mathcal{T}$ y algún $d \in \Delta^{\mathcal{I}}$ con $d \in C^{\mathcal{I}}$;*

II) *C está* subsumido por D con respecto a $\mathcal{T}$, *escrito $\mathcal{T} \vDash C \sqsubseteq D$, si y solo si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$;*

III) *C y D son* equivalentes con respecto a $\mathcal{T}$, *escrito $\mathcal{T} \vDash C \equiv D$, si y solo si $C^{\mathcal{I}} = D^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$;*

IV) *$\mathcal{K}$ es* consistente *si existe un modelo de $\mathcal{K}$;*

V) *b es una* instancia de C con respecto a $\mathcal{K}$, *escrito $\mathcal{K} \vDash b : C$, si y solo si $b^{\mathcal{I}} \in C^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{K}$.*

[3]Del inglés *Open World Assumption* que supone que no se dispone de un conocimiento perfecto o total del entorno, y por tanto, lo que se desconoce no es necesariamente falso.

[4]Del inglés *Closed World Assumption* que supone, en contraste con la Hipótesis de mundo abierto, que todo lo que no figura en la base se le otorga valor de verdad falso.

VI) *C es* satisfacible*, está* subsumido *por D o es* equivalente *a D si y solo si se cumple dicha propiedad respecto a la TBox vacía $\mathcal{T} = \emptyset$. C es* insatisfacible *si no es satisfacible respecto a ninguna TBox, es decir, si no es satisfacible respecto a la TBox vacía.*

Usaremos el símbolo $\models$, habitualmente utilizado para denotar consecuencia en la lógica de primer orden, para resaltar la coincidencia semántica con este concepto. Por otro lado, para recalcar que, una vez que $\mathcal{T}$ ha sido fijada, estar subsumido y la equivalencia entre dos conceptos es una relación binaria, denotaremos $C \sqsubseteq_{\mathcal{T}} D$ para II) y $C \equiv_{\mathcal{T}} D$ para III).

Tenemos que I) y II) están definidas con respecto a una TBox mientras que IV) y V) están definidas respecto a una ontología formada por una TBox y una ABox. Si queremos estudiar estas últimas nociones sobre una TBox (o una ABox), podemos extender dicha TBox (respectivamente ABox) a una ontología tomando la ABox (respectivamente, TBox) vacía; así no añadimos ninguna restricción dado que todas las interpretaciones posibles son modelos de una ABox (o TBox) vacía. Es decir, la consistencia de una TBox $\mathcal{T}$ (o de una ABox $\mathcal{A}$) es equivalente a la consistencia de $\mathcal{K} = (\mathcal{T}, \emptyset)$ (o $\mathcal{K} = (\emptyset, \mathcal{A})$) y que b sea una instancia de C con respecto a una TBox $\mathcal{T}$ (o una ABox $\mathcal{A}$) es equivalente a que sea una instancia de C con respecto a $\mathcal{K} = (\mathcal{T}, \emptyset)$ (o $\mathcal{K} = (\emptyset, \mathcal{A})$).

Nótese la diferencia entre que un elemento esté *en la extensión de* un concepto C en una interpretación $\mathcal{I}$ y que un individuo sea *una instancia de* un concepto C. Un individuo puede estar en la extensión de un concepto para una interpretación concreta $\mathcal{I}$, como el caso de *tortilla1* en Vegetariana visto en la página 15, sin ser necesariamente una instancia de dicho concepto. Para darse este último caso una base de conocimiento ha de obligar a que en todos sus modelos la interpretación del nombre del individuo esté siempre en la extensión del concepto, bien mediante un aserto explícito, por ejemplo $tortilla1$: Vegetariana, bien porque puede deducirse a partir de los demás asertos y axiomas.

En la Sección 3.3 hemos visto que $\mathcal{I}'$ es un modelo de la base de conocimiento $\mathcal{K}_e = (\mathcal{T}_e, \mathcal{A}_e)$, luego $\mathcal{K}_e$ es consistente. Tenemos que $juan$ es una instancia de Cocinero respecto a $\mathcal{K}_e$ ya que, aunque no lo hayamos definido como tal en nuestra ABox, sabemos por los asertos ($\mathcal{A}_e$.1), ($\mathcal{A}_e$.8) y ($\mathcal{A}_e$.9) que es una persona y cocina algunos platos. Por lo tanto, usando el axioma ($\mathcal{T}_e$.2) de nuestra TBox (que dice que todo aquello que tenga un *cocina-valor* es un cocinero) deducimos que en todo modelo $\mathcal{I}$ de $\mathcal{K}_e$, $juan^{\mathcal{I}} \in \mathsf{Cocinero}^{\mathcal{I}}$.

Casi todos los conceptos que hemos visto en el desarrollo del trabajo son satisfacibles pero podemos encontrar ejemplos de insatisfabilidad. Por ejemplo, $A \sqcap \neg A$ es insatisfacible, ya que si hubiese una interpretación $\mathcal{I}$ tal que un elemento b del dominio $\Delta^{\mathcal{I}}$ estuviese en la extensión del concepto, tendríamos que $b \in (A \sqcap \neg A)^{\mathcal{I}} = \emptyset$ por el Lema 2.3, lo que nos lleva a contradicción.

Nótese también que puede haber conceptos que sean satisfacibles en alguna TBox pero no en todas. Volviendo a la TBox de la Figura 3.1 podemos ver que el concepto $\mathsf{Comida} \sqcap \exists cocina.\mathsf{Comida}$ es insatisfacible con respecto a $\mathcal{T}_e$ ya que por ($\mathcal{T}_e$.1) sabemos que $\mathsf{Cocinero} \sqsubseteq \mathsf{Persona}$ y usando ($\mathcal{T}_e$.5) tenemos que $\mathsf{Cocinero} \sqcap \mathsf{Comida} = \bot$. Pero por ($\mathcal{T}_e$.2) $\exists cocina.\top \sqsubseteq \mathsf{Cocinero}$, lo que implica que nuestro concepto es insatisfacible con respecto a la TBox $\mathcal{T}_e$. Sin embargo, es fácil encontrar una TBox respecto a la cual sea un concepto satisfacible. Por ejemplo, si $\mathcal{T} = \emptyset$ e $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ con $\Delta^{\mathcal{I}} = \{a, b\}$, $\mathsf{Comida}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ y $cocina^{\mathcal{I}} = \{(a, b)\}$, claramente $a^{\mathcal{I}} \in (\mathsf{Comida} \sqcap \exists cocina.\mathsf{Comida})^{\mathcal{I}}$ y por tanto es satisfacible.

A continuación exponemos algunas propiedades de la relación de subsunción.

Lema 3.7. *Sean C, D y E conceptos, b un individuo concreto y $(\mathcal{T}, \mathcal{A})$, $(\mathcal{T}', \mathcal{A}')$ bases de conocimiento con $\mathcal{T} \subseteq \mathcal{T}'$ y $\mathcal{A} \subseteq \mathcal{A}'$. Entonces:*

I) $C \sqsubseteq_{\mathcal{T}} C$

II) *Si* $C \sqsubseteq_{\mathcal{T}} D$ *y* $D \sqsubseteq_{\mathcal{T}} E$ *entonces* $C \sqsubseteq_{\mathcal{T}} E$.

III) *Si b es una instancia de C con respecto a $(\mathcal{T}, \mathcal{A})$ y $C \sqsubseteq_{\mathcal{T}} D$, entonces b es una instancia de D con respecto a $(\mathcal{T}, \mathcal{A})$.*

IV) *Si* $\mathcal{T} \models C \sqsubseteq D$ *entonces* $\mathcal{T}' \models C \sqsubseteq D$.

V) *Si* $\mathcal{T} \models C \equiv D$ *entonces* $\mathcal{T}' \models C \equiv D$.

VI) *Si* $(\mathcal{T}, \mathcal{A}) \models b : E$ *entonces* $(\mathcal{T}', \mathcal{A}') \models b : E$.

Demostración. Las propiedades I) a III) se derivan directamente de la definición de subsunción e instancia, junto con las propiedades de inclusión de conjuntos.

Las propiedades IV) a VI) son consecuencia directa de los Lemas 3.2 y 3.4 puesto que $\mathcal{T} \sqsubseteq \mathcal{T}'$ y $\mathcal{A} \sqsubseteq \mathcal{A}'$. □

Concluimos que la relación de subsumición es reflexiva y transitiva por las propiedades I) y II); es decir, se trata de una relación de preorden. Por otro lado, con las propiedades IV)-VI) se prueba que las bases de conocimiento $\mathcal{ALC}$ son *monótonas*: cuantos más axiomas y asertos tiene la base de conocimiento, más consecuencias lógicas tendremos.

Estamos ahora en condiciones de ampliar los resultados del Lema 2.3, donde vimos que el conjunto de constructores de $\mathcal{ALC}$ no es minimal, en el sentido de que podemos prescindir de algunos de los constructores sin perder expresividad.

Lema 3.8. *Sean C y D conceptos, r un rol, $\mathcal{T}_0 = \emptyset$ la TBox vacía y $\mathcal{T}$ una TBox cualquiera. Entonces:*

I) $\mathcal{T}_0 \models \top \equiv (\neg C \sqcup C)$.

II) $\mathcal{T}_0 \models \bot \equiv (\neg C \sqcap C)$.

III) $\mathcal{T}_0 \models C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$.

IV) $\mathcal{T}_0 \models \forall r.C \equiv \neg(\exists r.\neg C)$.

V) $\mathcal{T} \models C \sqsubseteq D$ *si y solo si* $\mathcal{T} \models \top \sqsubseteq (\neg C \sqcup D)$.

Demostración. Las equivalencias I) y II) son consecuencia directa de las dos primeras propiedades del Lema 2.3.

Para III) usamos las propiedades III) y V) del Lema 2.3 de modo que para toda interpretación $\mathcal{I}$, $(C \sqcup D)^{\mathcal{I}} = (\neg\neg(C \sqcup D))^{\mathcal{I}} = (\neg(\neg C \sqcap \neg D))^{\mathcal{I}}$, y por tanto tenemos la propiedad deseada.

Para IV) por las propiedades III) y VII) del Lema 2.3, tenemos que para toda interpretación $\mathcal{I}$, $(\forall r.C)^{\mathcal{I}} = (\neg\neg(\forall r.C))^{\mathcal{I}} = (\neg(\exists r.\neg C))^{\mathcal{I}}$.

Para v), supongamos que $\mathcal{T} \vDash C \sqsubseteq D$ y sea $\mathcal{I}$ un modelo de $\mathcal{T}$; tenemos que $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Sea $a \in \Delta^{\mathcal{I}}$ tenemos dos casos: $a \in C^{\mathcal{I}}$ entonces, por lo visto anteriormente, tenemos que $a \in D^{\mathcal{I}}$ y por lo tanto, $a \in (\neg C \sqcup D)^{\mathcal{I}}$; $a \notin C^{\mathcal{I}}$ y directamente ya tenemos que $a \in (\neg C \sqcup D)^{\mathcal{I}}$. Por lo tanto tenemos que $\mathcal{T} \vDash \top \sqsubseteq (\neg C \sqcup D)$. El inverso es análogo. □

Como mencionábamos antes, usando este lema podemos reescribir los conceptos como otros equivalentes sin usar $\top$, $\bot$, la disyunción ni la restricción universal. También se podrían reescribir las propiedades del Lema 3.7 para prescindir de la conjunción en lugar de la disyunción y de la restricción universal en vez de la existencial. Como consecuencia añadida del Lema 3.7, puesto que la TBox vacía está contenida en todas las demás, tenemos que las cuatro primeras equivalencias se cumplen respecto a todas las TBoxes. Equivalencias con esta propiedad se denominan *tautologías*.

Los problemas de razonamiento de la Definición 3.6 pueden parecer independientes, pero en realidad se reducen a uno solo, como se recoge en el siguiente teorema.

Teorema 3.9. *Sean $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ una base de conocimiento $\mathcal{ALC}$, C y D conceptos y b un individuo concreto. Entonces:*

i) *$C \equiv_{\mathcal{T}} D$ si y sólo si $C \sqsubseteq_{\mathcal{T}} D$ y $D \sqsubseteq_{\mathcal{T}} C$.*

ii) *$C \sqsubseteq_{\mathcal{T}} D$ si y solo si $C \sqcap \neg D$ no es satisfacible con respecto a $\mathcal{T}$.*

iii) *C es satisfacible con respecto a $\mathcal{T}$ si y sólo si $C \not\sqsubseteq_{\mathcal{T}} \bot$.*

iv) *C es satisfacible con respecto a $\mathcal{T}$ si y sólo si $(\mathcal{T}, \{b : C\})$ es consistente.*

v) *$(\mathcal{T}, \mathcal{A}) \vDash b : C$ si y sólo si $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$ es no consistente.*

Demostración. i) Supongamos que $C \equiv_{\mathcal{T}} D$. Por definición, esto significa que $C^{\mathcal{I}} = D^{\mathcal{I}}$ para cada modelo $\mathcal{I}$ de $\mathcal{T}$. Por lo tanto $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ y $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$ y en consecuencia $C \sqsubseteq_{\mathcal{T}} D$ y $D \sqsubseteq_{\mathcal{T}} C$. El recíproco es análogo.

ii) Supongamos que $C \sqsubseteq_{\mathcal{T}} D$. Por definición, $C^{\mathcal{I}} = D^{\mathcal{I}}$ para cada modelo $\mathcal{I}$ de $\mathcal{T}$. Por lo tanto, no existe ningún modelo $\mathcal{I}$ tal que existe $x \in \Delta^{\mathcal{I}}$ con $x \in C^{\mathcal{I}}$ y $x \notin D^{\mathcal{I}}$, o lo que es equivalente, $x \in C^{\mathcal{I}} \cap (\neg D)^{\mathcal{I}}$, por lo tanto $C \sqcap \neg D$ no es satisfacible con respecto a $\mathcal{T}$. En la otra dirección tenemos que si $C \sqcap \neg D$ no es satisfacible con respecto a $\mathcal{T}$, entonces para todo modelo $\mathcal{I}$ de $\mathcal{T}$ $(C \sqcap \neg D)^{\mathcal{I}} = \emptyset$, o equivalente, $C^{\mathcal{I}} \cap \neg D^{\mathcal{I}} = \emptyset$. Ahora, dado $x \in C^{\mathcal{I}}$, debe cumplirse $x \in D^{\mathcal{I}}$, ya que si no fuese así tendríamos $x \in (\neg D)^{\mathcal{I}}$ y por lo tanto $x \in C^{\mathcal{I}} \cap \neg D^{\mathcal{I}} = \emptyset$, que nos lleva a contradicción. Por lo tanto $x \in D^{\mathcal{I}}$, con lo que llegamos a $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

iii) Sabemos que $\bot^{\mathcal{I}} = \emptyset$ en toda interpretación. Supongamos que C es satisfacible con respecto a $\mathcal{T}$, entonces existe un modelo $\mathcal{I}$ de $\mathcal{T}$ con $C^{\mathcal{I}} \neq \emptyset$, y por lo tanto $C^{\mathcal{I}} \not\subseteq \bot^{\mathcal{I}}$, con lo que $C \not\sqsubseteq \bot$. En la otra dirección, por contrarrecíproco, si C no es satisfacible con respecto a $\mathcal{T}$, entonces para todo modelo $\mathcal{I}$ de $\mathcal{T}$ tenemos que $C^{\mathcal{I}} = \emptyset$ y por lo tanto $C \sqsubseteq_{\mathcal{T}} \bot$.

iv) Sea C satisfacible respecto a $\mathcal{T}$. Entonces existe algún modelo $\mathcal{I}$ de $\mathcal{T}$ con $C^{\mathcal{I}} \neq \emptyset$. Tomamos un $x \in C^{\mathcal{I}}$ y extendemos $\mathcal{I}$ incluyendo $b^{\mathcal{I}} = x$. Es claro que $\mathcal{I}$ sigue siendo un modelo de $\mathcal{T}$ y, además, hace de $\mathcal{I}$ un modelo de la ABox $\{b : C\}$. Por lo tanto la base de conocimiento $(\mathcal{T}, \{b : C\})$ es consistente. El recíproco es trivial, ya que si $(\mathcal{T}, \{b : C\})$ es consistente, existe un modelo $\mathcal{I}$ que cumple que $b^{\mathcal{I}} \in C^{\mathcal{I}}$ y por lo tanto $C^{\mathcal{I}} \neq \emptyset$ y C es satisfacible con respecto a $\mathcal{T}$.

v) Sea b una instancia de C con respecto a $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Por definición $b^{\mathcal{I}} \in C^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{K}$. Puesto que $C^{\mathcal{I}}$ y $(\neg C)^{\mathcal{I}}$ son disjuntos, no puede existir ningún modelo $\mathcal{I}$ de $(\mathcal{T}, \mathcal{A})$ tal que $b^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$ y por lo tanto la base de conocimiento extendida $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$ no es consistente. En el otro sentido, si suponemos que $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$ es inconsistente, tenemos dos casos posibles: $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ inconsistente o $\mathcal{K}$ consistente. El primer caso es trivial ya que si es inconsistente no existen modelos de la misma y, por lo tanto, no es posible encontrar un modelo de $\mathcal{K}$ para el que no se cumpla una propiedad cualquiera. En el segundo caso, si existiese un modelo $\mathcal{I}$ de $\mathcal{K}$ tal que $b^{\mathcal{I}} \in (\neg C)^{\mathcal{I}}$ contradiríamos la hipótesis de inconsistencia de $(\mathcal{T}, \mathcal{A} \cup \{b : \neg C\})$. Por lo tanto no existe tal modelo, es decir que $b^{\mathcal{I}} \in C^{\mathcal{I}}$ para todos los modelos de $\mathcal{K}$, siendo así b una instancia de C con respecto a $\mathcal{K}$.

□

Como consecuencia de este teorema, podemos centrarnos en resolver el problema de la consistencia, ya que el resto de problemas de razonamiento pueden ser reducidos a este mismo. Por lo tanto, si tenemos un algoritmo para decidir sobre la consistencia de una base de conocimiento, podremos decidir sobre los demás problemas. En el Capítulo 5 describiremos un método para resolver el problema de consistencia conocido como Tableaux Semánticos y demostraremos su corrección y completud. Para ello previamente introduciremos algunos conceptos de Teoría de Modelos en el Capítulo 4.

3.5. $\mathcal{ALC}$ como fragmento decidible de la lógica de primer orden

En esta sección se expondrá la relación existente entre la lógica descriptiva $\mathcal{ALC}$ y la lógica de primer orden. En particular, veremos cómo pueden identificarse los nombres de roles y de conceptos con predicados binarios y unarios respectivamente, de modo que $\mathcal{ALC}$ se corresponde con un fragmento decidible de la lógica de predicados.

Dados un axioma de equivalencia de una TBox, un axioma de una TBox o un aserto de ABox podemos expresarlos en lógica de primer orden usando dos variables x e y. Por ejemplo, si tenemos:

$$\begin{aligned} \exists cocina.\top &\sqsubseteq \mathsf{Persona} \\ \mathsf{Cocinero} &\equiv \mathsf{Persona} \sqcap \exists cocina.\mathsf{Comida} \\ jose : &\ \mathsf{Cocinero} \end{aligned}$$

podemos traducirlo directamente a:

$$\begin{aligned} &\forall x(\exists y cocina(x, y) \Rightarrow \mathsf{Persona}(x)) \\ &\forall x(\mathsf{Cocinero}(x) \Leftrightarrow \mathsf{Persona}(x) \wedge \exists y.(cocina(x, y) \wedge \mathsf{Comida}(y))) \\ &\mathsf{Cocinero}(jose) \end{aligned}$$

En general, definimos dos aplicaciones π_x y π_y que hacen corresponder a cada concepto $\mathcal{ALC}$ con una fórmula de lógica de primer orden con una variable libre, ya sea x o y, como sigue:

$$\begin{aligned}
\pi_x(A) &= A(x), & \pi_y(A) &= A(y),\\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D), & \pi_y(C \sqcap D) &= \pi_y(C) \wedge \pi_y(D),\\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D), & \pi_y(C \sqcup D) &= \pi_y(C) \vee \pi_y(D),\\
\pi_x(\exists r.C) &= \exists y(r(x,y) \wedge \pi_y(C)), & \pi_y(\exists r.C) &= \exists x(r(y,x) \wedge \pi_x(C)),\\
\pi_x(\forall r.C) &= \forall y(r(x,y) \Rightarrow \pi_y(C)), & \pi_y(\forall r.C) &= \forall x(r(y,x) \Rightarrow \pi_x(C)).
\end{aligned}$$

Una vez definidas estas funciones, podemos traducir una TBox $\mathcal{T}$ y una ABox $\mathcal{A}$ mediante la siguiente función de traducción π, donde, dada Ψ una fórmula de lógica de primer orden, $\Psi[x \mapsto a]$ es la fórmula obtenida al sustituir x por a en Ψ:

$$\pi(\mathcal{T}) = \forall x \bigwedge_{C \sqsubseteq D \in \mathcal{T}} (\pi_x(C) \Rightarrow \pi_x(D)),$$

$$\pi(\mathcal{A}) = \bigwedge_{a:C \in \mathcal{A}} (\pi_x(C)[x \mapsto a] \wedge \bigwedge_{(a,b):r \in \mathcal{A}} r(a,b).$$

Claramente, esta traducción conserva la semántica de $\mathcal{ALC}$ de la Definición 2.2, y por tanto de los problemas de razonamiento de la Definición 3.6.[5]

Teorema 3.10. *Sean $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ una base de conocimiento en $\mathcal{ALC}$, C, D conceptos $\mathcal{ALC}$ y b un individuo concreto. Entonces:*

I) *$\mathcal{K} = (\mathcal{T}, \mathcal{A})$ es satisfacible si y sólo si $\pi(\mathcal{T}) \wedge \pi(\mathcal{A})$ es satisfacible,*

II) *$C \sqsubseteq_{\mathcal{T}} D$ si y sólo si $\pi(\mathcal{T}) \Rightarrow \forall x(\pi_x(C) \Rightarrow \pi_x(D))$ es válido,*

III) *b es una instancia de C con respecto a $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ si y sólo si $(\pi(\mathcal{T}) \wedge \pi(\mathcal{A})) \Rightarrow \pi_x(C)[x \mapsto b]$ es válida.*

Con este teorema podemos traducir todo problema de razonamiento expuesto en la Definición 3.6 a un problema de decidir sobre la satisfabilidad de una fórmula en lógica de primer orden con dos variables. En [7] se demuestra que FO^2, el fragmento de la lógica de primer orden formado por fórmulas escritas con dos variables, es decidible y que en este fragmento el problema de satisfabilidad es *EXPTIME-completo*. En consecuencia, cada uno de estos problemas de razonamiento son decidibles.

3.6. Extensiones de $\mathcal{ALC}$

En ocasiones, $\mathcal{ALC}$ carece de suficiente expresividad, de modo que se hace necesario enriquecer este lenguaje básico con nuevos operadores. A continuación motivamos y definimos algunos de los más habituales.

[5] La demostración formal de la correspondencia entre la semántica y los problemas de razonamiento en ambas lógicas se realiza mediante inducción estructural. Al tratarse de un resultado secundario la omitimos de esta memoria dada su extensión, pues exigiría introducir la notación y definir formalmente la sintaxis y semántica de la lógica de predicados. El lector interesado puede consultar en [4].

3.6.1. Roles inversos

En nuestro ejemplo del restaurante, podemos en un momento determinado querer diferenciar la comida que se sirve sin cocinar, por ejemplo una pieza de fruta, de la que se sirve cocinada. Para ello podemos identificar este último tipo de comida con el concepto compuesto $\mathsf{Comida} \sqcap \exists \mathit{cocinado\text{-}por}.\top$ donde el rol *cocinado-por* correspondería al inverso de *cocina*. Teniendo la información del ejemplo de la Figura 2.2, sabiendo que *juan cocina el-huevo*, lo lógico sería deducir que se cumple la relación *el-huevo cocinado-por juan* y, por tanto, *el-huevo* está en el concepto compuesto que hemos definido. Pero no es así debido a que tratamos los roles *cocina* y *cocinado-por* como independientes. Deberíamos poder identificar uno como rol inverso del otro, con esta motivación surgen los roles inversos.

Añadiendo roles inversos a $\mathcal{ALC}$ damos lugar a $\mathcal{ALCI}$, añadiendo un nuevo constructor a la sintaxis y definiendo su semántica.

Definición 3.11. *Sea* $\mathbf{R}$ *el conjunto de nombre de roles; para cada nombre de rol* $r \in \mathbf{R}$ *denotamos como* r^- *a su* rol inverso*, de manera que el conjunto de* $\mathcal{I}$*-roles viene dado por* $\mathbf{R} \cup \{r^- : r \in \mathbf{R}\}$*. Sea* $\mathcal{L}$ *una lógica descriptiva, el conjunto de* conceptos $\mathcal{LI}$ *será el menor conjunto que contenga a los conceptos de* $\mathcal{L}$ *junto con los obtenidos al considerar los* $\mathcal{I}$*-roles en lugar de solamente los roles de* $\mathbf{R}$*.*

Para dotar a los conceptos $\mathcal{LI}$ de una semántica, ampliamos la interpretación de la semántica de $\mathcal{L}$ a los roles inversos como sigue:

Definición 3.12. *Siendo* r *un nombre de rol, una interpretación* $\mathcal{I}$ *lleva roles inversos a relaciones binarias en* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *como sigue:*

$$(r^-)^{\mathcal{I}} = \{(y, x) : (x, y) \in r^{\mathcal{I}}\}$$

De esta definición se deduce trivialmente que $(x, y) \in r^{\mathcal{I}}$ si y sólo si $(y, x) \in (r^-)^{\mathcal{I}}$. Ahora, continuando con el ejemplo anterior, es tan fácil como redefinir el concepto de la comida cocinada como $\mathsf{Comida} \sqcap \exists \mathit{cocina}^-.\top$ y así tendríamos directamente que *el-huevo* pertenece a este concepto que hemos definido.

3.6.2. Restricciones numéricas

Para describir un concepto, podría ocurrir que queramos restringir el número de elementos con los que se relaciona a través de un rol. Este podría ser el caso cuando queremos, por ejemplo, identificar a un catador: podríamos suponer para identificarlo que será un comensal que consume más de 10 platos diferentes. Sin embargo, para expresar este concepto no tenemos herramientas suficientes con los operadores de $\mathcal{ALC}$. Tenemos entonces que definir las llamadas restricciones numerales y las restricciones numerales calificativas[6].

Definición 3.13. *Sean* $n \in \mathbb{N}$*,* r *un rol y* C *un concepto. Decimos que una* restricción numérica *es el concepto representado por* $(\leq n\ r)$ *o* $(\geq n\ r)$*, y decimos que una* restricción numérica calificativa *es el concepto representado por* $(\leq n\ r.C)$ $(\geq n\ r.C)$ *donde* C *se denomina* concepto calificativo*. Dada* $\mathcal{L}$ *una lógica descriptiva, la lógica* $\mathcal{LN}$ *o* $\mathcal{LQ}$ *se obtiene añadiendo a* $\mathcal{L}$ *los constructores de restricción numérica o restricción numérica calificativa respectivamente.*

[6]Del inglés *qualified number restrictions*.

Para dotar de una semántica a estos conceptos debemos ampliar la interpretación de la semántica de $\mathcal{L}$ como sigue:

Definición 3.14. *Sean $n \in \mathbb{N}$, r un rol y C un concepto. Dada una interpretación $\mathcal{I}$, se amplía a las restricciones numéricas como sigue:*

$$\begin{aligned}(\leq n\ r)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : |\{e : (d,e) \in r^{\mathcal{I}}\}| \leq n\} \\ (\geq n\ r)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : |\{e : (d,e) \in r^{\mathcal{I}}\}| \geq n\} \\ (\leq n\ r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : |\{e : (d,e) \in r^{\mathcal{I}}, e \in C^{\mathcal{I}}\}| \leq n\} \\ (\geq n\ r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : |\{e : (d,e) \in r^{\mathcal{I}}, e \in C^{\mathcal{I}}\}| \geq n\}\end{aligned}$$

donde $|S|$ denota el cardinal del conjunto S.

Se puede abreviar la restricción $(\leq\ n\ r) \sqcap (\geq\ n\ r)$ de la forma $(=\ n\ r)$, y de la misma forma $(=\ n\ r.C)$ es una abreviatura de $(\leq\ n\ r.C) \sqcap (\geq\ n\ r.C)$. Nótese que la restricción numérica calificativa $(\geq\ 1\ r.C)$ es equivalente al concepto $\exists r.C$ y por otro lado cualquier restricción numérica sobre un rol $(\geq\ n\ r)$ es equivalente a la restricción numérica calificativa $(\geq\ n\ r.C)$ con $C = \top$. Con estos constructores podemos restringir la cantidad de *r-valores* que puede tener un elemento del concepto definido y, en el caso de las calificativas, la cantidad de *r-valores* que están en la extensión del concepto C. Así, en el ejemplo anterior podríamos describir al comensal que consume al menos 10 platos como $\mathsf{Comensal} \sqcap (\geq 10\, \mathit{consume}.\mathsf{Comida})$.

Añadiendo las restricciones numéricas (o las restricciones numéricas calificativas) a $\mathcal{ALC}$ obtenemos respectivamente la lógica descriptiva $\mathcal{ALCN}$ (o $\mathcal{ALCQ}$).

3.6.3. Nominales

En ocasiones, hay conceptos que dependen de un nombre de individuo. Sin embargo, tal y cómo tenemos definida la sintaxis de $\mathcal{ALC}$ no podemos, por ejemplo, definir un concepto para referirnos a aquellos platos que son especialidad (y por tanto, son cocinados) por un cocinero concreto, *juan*. Así surgen los nominales.

Definición 3.15. *Sea $b \in \mathbf{I}$ un individuo concreto; definimos el* nominal *de b como el concepto representado por $\{b\}$. Dada $\mathcal{L}$ una lógica descriptiva, denotamos $\mathcal{LO}$ a la lógica que se obtiene añadiendo el constructor de concepto nominal a $\mathcal{L}$.*

Para definir la semántica de $\mathcal{LO}$ basta extender las interpretaciones a nominales como sigue.

Definición 3.16. *Dada una interpretación $\mathcal{I}$, esta interpretación se extiende a los conceptos nominales de la siguiente forma:*

$$b^{\mathcal{I}} \in \Delta^{\mathcal{I}} \quad (\{b\})^{\mathcal{I}} = \{b^{\mathcal{I}}\}$$

Los nominales nos permitirán referirnos a elementos concretos como conceptos y trabajar con ellos de la misma manera que con un concepto simple. Esto aporta mayor

expresividad a nuestra lógica. Así, podemos representar los platos cocinados por Juan como $\mathsf{Comida} \sqcap \forall cocina^{-}.\{juan\}$ que, con la interpretación del ejemplo de la Figura 2.2 tendríamos que son {*la-ensalada*, *la-tortilla* y *el-huevo*}. Nótese que este concepto estaría en la lógica $\mathcal{ALC}$ extendida con nominales y roles inversos, es decir, $\mathcal{ALCOI}$.

3.6.4. Jerarquía de roles

Es común encontrarse con roles que están relacionados entre sí. Por ejemplo si alguien es especialista en un plato, entonces es él o ella quien lo prepara. Así, si sabemos que un repostero es especialista exclusivamente en los postres, deberíamos deducir que todo lo que cocina él o ella son postres. Pero, si no podemos modelar la relación entre estos dos roles, esta información pasa desapercibida ya que interpretamos nuestros roles (*especialista-en* y *cocina*) como independientes. Así surge la jerarquía de roles.

Definición 3.17. *Sean $r, s \in \mathbf{R}$ roles; definimos un* axioma de inclusión de roles *(RIA)*[7] *como un axioma de la forma $r \sqsubseteq s$. Si $\mathcal{L}$ es una lógica descriptiva, la lógica que resulta de añadir el axioma de inclusión de roles se denota $\mathcal{LH}$.*

Para dotar a los conceptos de $\mathcal{LH}$ de una semántica, ampliamos cualquier interpretación en $\mathcal{L}$ a la jerarquía como sigue:

Definición 3.18. *Dada una interpretación $\mathcal{I}$, para que los roles cumplan los axiomas de inclusión se tiene que dar:*

$$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$$

Gracias a la inclusión de roles, si dos elementos están relacionados por cierto rol y este se encuentra en algún *RIA*, entonces podremos deducir otras relaciones entre esos dos conceptos. En nuestro caso en concreto, si especificamos que *especialista-en* $\sqsubseteq$ *cocina* en la TBox, entonces al decir que el repostero es especialista exclusivamente en postres $\mathsf{Repostero} \sqsubseteq \forall especialista\text{-}en.\mathsf{Postre}$, deduciremos inmediatamente que todo lo que cocina son postres, es decir, $\mathsf{Repostero} \sqsubseteq \forall cocina.\mathsf{Postre}$.

3.6.5. Roles transitivos

Si un rol representa una relación transitiva, es natural querer modelar esta situación para sacarle provecho. Como ejemplo, el rol *contiene* puede ser esencial a la hora de conocer información importante para alérgenos. Si sabemos que Bechamel *contiene* Leche y por otro lado, que Lasaña *contiene* Bechamel, sería esencial deducir la relación Lasaña *contiene* Leche pero, tal y como están definidos los constructores en $\mathcal{ALC}$ entre Lasaña y Leche no hay ninguna relación. Así, se introduce la transitividad de roles:

Definición 3.19. *Sea $r \in \mathbf{R}$ un rol; un* axioma de transitividad de roles *es una expresión de la forma $Trans(r)$. Para denotar que a una lógica descriptiva $\mathcal{L}$ se le añade el axioma de transitividad, se sustituye en el nombre las letras $\mathcal{ALC}$ por $\mathcal{S}$ o, alternativamente, se le añade el subíndice $\mathcal{R}^{+}$*

[7] Del inglés *Role Inclusion Axiom*.

La lógica $\mathcal{ALCOI}$ con nominales y roles inversos extendida con roles transitivos se denota $\mathcal{SOI}$ o alternativamente, $\mathcal{ALCOI}_{\mathcal{R}^+}$ (aunque esta última notación es menos frecuente).

Definición 3.20. *Dada una interpretación $\mathcal{I}$, para que un rol cumpla los axiomas de transitividad $r^{\mathcal{I}}$ ha de ser una relación transitiva, es decir, si $a, b, c \in \mathbf{I}$ son nombres de individuos:*

$$si\ (a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}\ y\ (b^{\mathcal{I}}, c^{\mathcal{I}}) \in r^{\mathcal{I}},\ entonces\ (a^{\mathcal{I}}, c^{\mathcal{I}}) \in r^{\mathcal{I}}$$

Añadiendo el axioma de transitividad al rol *contiene*, es decir, añadiendo $Trans(contiene)$ a la TBox, si Lasaña *contiene* Bechamel y que Bechamel *contiene* Leche, entonces tendremos directamente que Lasaña *contiene* Leche. Así podemos deducir información sobre alérgenos aunque no la hayamos incluido explícitamente en la base de conocimiento.

Todas las extensiones descritas conllevan un aumento de la expresividad de nuestra LD, pero a la par aumenta el coste computacional de los problemas de razonamiento[8].

[8]La complejidad según los constructores utilizados en la LD (si se conoce) puede consultarse en el navegador de complejidad `www.cs.man.ac.uk/~ezolin/dl`.

Capítulo 4

Breve introducción a la teoría de modelos

La teoría de modelos se ocupa de estudiar relaciones entre un conjunto de fórmulas o axiomas (en nuestro caso, una ontología) y las interpretaciones que lo satisfacen. En este capítulo se introducirán algunos conceptos y propiedades de esta teoría necesarios para entender y estudiar el método de razonamiento del siguiente capítulo.

4.1. Tamaño de una base de conocimiento $\mathcal{ALC}$.

Intuitivamente, podemos medir el tamaño de un concepto en función de las veces que se han aplicado los constructores sintácticos para formarlo.

Definición 4.1. *Dado C un concepto $\mathcal{ALC}$, definimos el* tamaño *de C, denotado $size(C)$ y el conjunto de sus* subconceptos*, denotado $sub(C)$ por inducción en la estructura de C como sigue:*

- *Si C es un concepto simple, entonces $size(C) = 1$ y $sub(C) = \{C\}$.*
- *Si $C = C_1 \sqcap C_2$ o $C = C_1 \sqcup C_2$ con C_1 y C_2 conceptos $\mathcal{ALC}$, entonces $size(C) = 1 + size(C_1) + size(C_2)$ y $sub(C) = \{C\} \cup sub(C_1) \cup sub(C_2)$.*
- *Si $C = \neg D$ o $C = \exists r.D$ o $C = \forall r.D$ con D concepto $\mathcal{ALC}$, entonces $size(C) = 1 + size(D)$ y $sub(C) = \{C\} \cup sub(D)$.*

El tamaño cuenta el número de conceptos simples (incluyendo $\top$ y $\bot$), el número de roles y el número de operadores dentro de un concepto. Así tendríamos que

$$size(A \sqcap \exists r.(A \sqcup B)) = 1 + 1 + (1 + (1 + 1 + 1)) = 6$$

A su vez, el conjunto de subconceptos incluye todos los conceptos que se han ido formando durante la construcción sintáctica de C. Por ejemplo, tendríamos que

$$sub(A \sqcap \exists r.(A \sqcup B)) = \{A \sqcap \exists r.(A \sqcup B), A, \exists r.(A \sqcup B), A \sqcup B, B\}$$

Estas nociones se pueden extender con facilidad a una TBox, a una ABox y a una ontología como sigue:

Definición 4.2. *Dadas $\mathcal{T}$ una TBox y $\mathcal{A}$ una ABox definimos el* tamaño *y el* conjunto de subconceptos *de la TBox, ABox y de la ontología $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ como*

$$size(\mathcal{T}) = \sum_{C \sqsubset D \in \mathcal{T}} size(C) + size(D) \text{ y } sub(\mathcal{T}) = \bigcup_{C \sqsubset D \in \mathcal{T}} sub(C) \cup sub(D)$$

$$size(\mathcal{A}) = \sum_{a:C \in \mathcal{A}} size(C) \text{ y } sub(\mathcal{A}) = \bigcup_{a:C \in \mathcal{A}} sub(C)$$

$$size(\mathcal{T}) = size(\mathcal{T}) + size(\mathcal{A}) \text{ y } sub(\mathcal{K}) = sub(\mathcal{T}) \cup sub(\mathcal{A})$$

Es fácil deducir el siguiente resultado:

Lema 4.3. *Sean C un concepto $\mathcal{ALC}$, $\mathcal{T}$ una TBox $\mathcal{ALC}$ y $\mathcal{A}$ una ABox $\mathcal{ALC}$. Entonces*

$$|sub(C)| \leq size(C) \text{ ,} |sub(\mathcal{T})| \leq size(\mathcal{T}) \text{ y } |sub(\mathcal{A})| \leq size(\mathcal{A})$$

donde $|\cdot|$ denota el cardinal de un conjunto.

Demostración. Veamos primero que se cumple la desigualdad para cualquier concepto C por inducción estructural:

- Si C es un concepto simple, tenemos que $|sub(C)| = 1 = size(C)$.
- Supongamos que se cumple la desigualdad para dos conceptos C_1, C_2 cualesquiera.
  - Si $C = C_1 \sqcap C_2$ o $C = C_1 \sqcup C_2$, entonces

    $|sub(C)| = |C \cup sub(C_1) \cup sub(C_2)| = 1 + |sub(C_1) \cup sub(C_2)| \leq 1 + |sub(C_1)| + |sub(C_2)|$

    y usando la hipótesis de inducción

    $$1 + |sub(C_1)| + |sub(C_2)| \leq 1 + size(C_1) + size(C_2)$$

    Pero por definición $size(C) = 1 + size(C_1) + size(C_2)$ y por tanto $|sub(C)| \leq size(C)$.
  - Si $C = \neg C_1$ o $C = \exists r.C_1$ o $C = \forall r.C_1$, entonces tenemos que

    $$|sub(C)| = 1 + |sub(C_1)| \leq 1 + size(C_1) = size(C_1),$$

    y por lo tanto, también se cumple la desigualdad.

Para la TBox $\mathcal{T}$ y la ABox $\mathcal{A}$ la desigualdad se obtiene directamente de la desigualdad para conceptos. □

Definición 4.4. *Sea S un conjunto de conceptos $\mathcal{ALC}$, decimos que S es* cerrado *si*

$$\bigcup \{sub(C) : C \in S\} \subseteq S$$

Claramente, si C es un concepto $\mathcal{ALC}$ y consideramos $S = sub(C)$ entonces S es cerrado.

Dado un conjunto cualquiera de conceptos S sus elementos pueden clasificarse desde un punto de vista semántico con la noción de S-tipo, intuitivamente, aquellos conceptos a los que pertenece un elemento según una interpretación.

Definición 4.5. *Dados S un conjunto de conceptos $\mathcal{ALC}$ y una interpretación $\mathcal{I}$, se define el* S-tipo *de un elemento $d \in \Delta^{\mathcal{I}}$ como:*

$$t_S(d) = \{C \in S : d \in C^{\mathcal{I}}\}$$

Puesto que un S-tipo es un subconjunto de S, hay como mucho tantos S-tipos como subconjuntos de S.

Lema 4.6. *Sean S un conjunto finito de conceptos $\mathcal{ALC}$ e $\mathcal{I}$ una interpretación. Entonces*

$$|\{t_S(d) : d \in \Delta^{\mathcal{I}}\}| \leq 2^{|S|}.$$

Nótese que, aunque es posible que tengamos varios o incluso infinitos elementos del dominio en un S-tipo, se trata de elementos con la misma propiedad. Así, si tomamos un "representante" de cada S-tipo puede obtenerse una nueva interpretación en cierto modo equivalente a la original pero cuyo dominio está acotado por $2^{|S|}$. Esta es la base para probar que todo concepto satisfacible dada una TBox tiene un modelo finito y, en consecuencia que el problema de satisfabilidad es decidible. Al estar fuera del ámbito de este TFG, omitiremos el desarrollo de estas ideas, pero el lector interesado puede consultar, por ejemplo, la sección 3.4 de [3].

4.2. Bisimulación

Formalizamos ahora la idea de interpretaciones "equivalentes" mediante la noción de bisimilitud. A lo largo de esta sección suponemos que los conjuntos de nombres $\mathbf{C}$ y $\mathbf{R}$ permanecen fijos.

Definición 4.7. *Sean $\mathcal{I}_1$ y $\mathcal{I}_2$ interpretaciones. La relación $\rho \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ es una* bisimulación *entre $\mathcal{I}_1$ y $\mathcal{I}_2$ si se cumplen las siguientes propiedades:*

1. *d_1 ρ d_2 implica que:*

   $$d_1 \in C^{\mathcal{I}_1} \text{ si y solo si } d_2 \in C^{\mathcal{I}_2}$$

   para todo $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$ y $C \in \mathbf{C}$;

2. *d_1 ρ d_2 y $(d_1, d_1') \in r^{\mathcal{I}_1}$ implica la existencia de $d_2' \in \Delta^{\mathcal{I}_2}$ tal que:*

   $$d_1' \ \rho \ d_2' \ y \ (d_2, d_2') \in r^{\mathcal{I}_2}$$

   para todo $d_1, d_1' \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$ y $r \in \mathbf{R}$;

3. *d_1 ρ d_2 y $(d_2, d_2') \in r^{\mathcal{I}_2}$ implica la existencia de $d_1' \in \Delta^{\mathcal{I}_1}$ tal que:*

   $$d_1' \ \rho \ d_2' \ y \ (d_1, d_1') \in r^{\mathcal{I}_1}$$

   para todo $d_1' \in \Delta^{\mathcal{I}_1}$, $d_2, d_2' \in \Delta^{\mathcal{I}_2}$ y $r \in \mathbf{R}$.

Dados $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$, denotamos:

$$(\mathcal{I}_1, d_1) \sim (\mathcal{I}_2, d_2)$$

si hay una bisimulación ρ entre $\mathcal{I}_1$ y $\mathcal{I}_2$ tal que d_1 ρ d_2 y diremos que $d_1 \in \mathcal{I}_1$ es bisimilar *a $d_2 \in \mathcal{I}_2$.*

Intuitivamente, d_1 y d_2 son bisimilares si están en las extensiones de los mismos conceptos y poseen las mismas relaciones con elementos que además son bisimilares entre sí.

Teorema 4.8. *Si $(\mathcal{I}_1, d_1) \sim (\mathcal{I}_2, d_2)$, entonces para todo C concepto $\mathcal{ALC}$ se cumple*

$$d_1 \in C^{\mathcal{I}_1} \text{ si y solo si } d_2 \in C^{\mathcal{I}_2}$$

Demostración. Si $(\mathcal{I}_1, d_1) \sim (\mathcal{I}_2, d_2)$, tenemos una bisimulación ρ entre $\mathcal{I}_1$ y $\mathcal{I}_2$ tal que d_1 ρ d_2. Probaremos el teorema por inducción en la estructura de C; gracias a la equivalencia de conceptos del Lema 3.8, basta considerar los constructores de conjunción, negación y restricción existencial.

- Caso base: si $C = A \in \mathbf{C}$, entonces

  $$d_1 \in A^{\mathcal{I}_1} \text{ si y solo si } d_2 \in A^{\mathcal{I}_2}$$

  como consecuencia inmediata de la primera propiedad de la definición de bisimulación d_1 ρ d_2.

- Caso inductivo: supongamos por hipótesis de inducción que la propiedad se cumple para todos los subconceptos de C.

  - Si $C = D \sqcap E$, entonces

    $$\begin{aligned} d_1 \in (D \sqcap E)^{\mathcal{I}_1} &\text{ si y sólo si } d_1 \in D^{\mathcal{I}_1} \text{ y } d_1 \in E^{\mathcal{I}_1}, && \text{(por la semántica de } \mathcal{ALC}) \\ &\text{si y sólo si } d_2 \in D^{\mathcal{I}_2} \text{ y } d_2 \in E^{\mathcal{I}_2}, && \text{(por hipótesis de inducción)} \\ &\text{si y sólo si } d_2 \in (D \sqcap E)^{\mathcal{I}_2}, && \text{(por la semántica de } \mathcal{ALC}) \end{aligned}$$

  - Si $C = \neg D$, entonces

    $$\begin{aligned} d_1 \in (\neg D)^{\mathcal{I}_1} &\text{ si y sólo si } d_1 \in \Delta^{\mathcal{I}_1} \setminus D^{\mathcal{I}_1}, && \text{(por la semántica de } \mathcal{ALC}) \\ &\text{si y sólo si } d_1 \notin D^{\mathcal{I}_1}, && \text{(por la semántica de } \mathcal{ALC}) \\ &\text{si y sólo si } d_2 \notin D^{\mathcal{I}_2}, && \text{(por hipótesis de inducción} \\ &\text{si y sólo si } d_2 \in \Delta^{\mathcal{I}_2} \setminus D^{\mathcal{I}_2}, && \text{(por la semántica de } \mathcal{ALC}) \\ &\text{si y sólo si } d_2 \in (\neg D)^{\mathcal{I}_2}, && \text{(por la semántica de } \mathcal{ALC}) \end{aligned}$$

  - Si $C = \exists r.D$, entonces

    $$\begin{aligned} d_1 \in (\exists r.D)^{\mathcal{I}_1} &\text{ si y sólo si existe } d_1' \in \Delta^{\mathcal{I}_1} \text{ tal que } (d_1, d_1') \in r^{\mathcal{I}_1} \text{ y } d_1' \in D^{\mathcal{I}_1}, \\ &\text{si y sólo si existe } d_2' \in \Delta^{\mathcal{I}_2} \text{ tal que } (d_2, d_2') \in r^{\mathcal{I}_2} \text{ y } d_2' \in D^{\mathcal{I}_2}, \\ &\text{si y sólo si } d_2 \in (\exists r.D)^{\mathcal{I}_2}, \end{aligned}$$

    donde la primera y la tercera equivalencia son debidas a la semántica de la restricción existencial y la segunda es debida a las condiciones 2ª y 3ª de la definición de bisimulación y a la hipótesis de inducción.

□

Este teorema implica que $\mathcal{ALC}$ no distingue entre conceptos bisimilares. Por otro lado podemos usar el concepto de bisimulación para comprobar que las ampliaciones de $\mathcal{ALC}$ expuestas en el Capítulo 3 son realmente ampliaciones, es decir, que existe algún concepto de la ampliación que no puede ser expresado con $\mathcal{ALC}$. Queda fuera del ámbito de este trabajo desarrollar estas ideas, pero el lector interesado puede consultar la sección 3.2 de [3].

4.3. Modelo de árbol

Terminamos nuestra breve incursión en la teoría de modelos demostrando que un concepto satisfacible posee un modelo en forma de árbol. Para ello recordamos la definición formal de árbol con raíz.

Definición 4.9. *Un* árbol *es un grafo dirigido $G = (V, E)$ tal que:*

- *V contiene una única* raíz*, es decir, un nodo $v_r \in V$ tal que no existe ningún nodo $v \in V$ con $(v, v_r) \in E$;*
- *todo nodo $v \in V \setminus \{v_r\}$ tiene un único predecesor, es decir, existe un único nodo $v' \in V$ tal que $(v', v) \in E$.*

En general, cualquier interpretación $\mathcal{I}$ puede representarse como un grafo dirigido $\mathcal{G}_\mathcal{I}$ donde los vértices corresponden a los elementos del dominio y con un arco dirigido etiquetado con un nombre de rol r si el elemento del vértice de origen está relacionado mediante r con el elemento del vértice destino en la interpretación, es decir

$$\mathcal{G}_\mathcal{I} = \left(\Delta^\mathcal{I}, \bigcup_{r \in \mathbf{R}} r^\mathcal{I} \right)$$

Gracias a su representación en forma de grafo, podemos definir la noción de d-camino en una interpretación.

Definición 4.10. *Sea una interpretación $\mathcal{I}$ y $d \in \Delta^\mathcal{I}$. Un* d-camino *en $\mathcal{I}$ es una secuencia finita $d_0, d_1, \ldots, d_{n-1}$ con $n \geq 1$ elementos de $\Delta^\mathcal{I}$ tal que:*

- *$d_0 = d$*
- *para todo $i, 1 \leq i < n$, existe un rol $r_i \in \mathbf{R}$ tal que $(d_{i-1}, d_i) \in {r_i}^\mathcal{I}$.*

Dado un d-camino $p = d_0, d_1, \ldots, d_{n-1}$, definimos su longitud como n y su nodo final como $end(p) = d_{n-1}$.

Formalizamos ahora la noción de modelo de un concepto con respecto a una TBox y modelo en árbol.

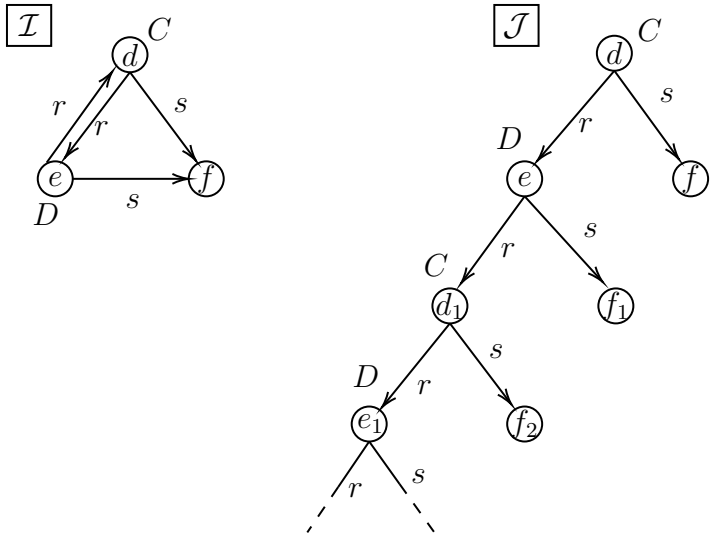

Figura 4.1: Modelo $\mathcal{I}$ y su desplegado $\mathcal{J}$.

Definición 4.11. *Una interpretación $\mathcal{I}$ es un* modelo de un concepto C con respecto a una TBox $\mathcal{T}$ *si $\mathcal{I}$ es un modelo de $\mathcal{T}$ tal que $C^{\mathcal{I}} \neq \emptyset$. Si $\Delta^{\mathcal{I}}$ es finito diremos que el modelo $\mathcal{I}$ es finito.*

Sea $\mathcal{T}$ una TBox $\mathcal{ALC}$ y C un concepto $\mathcal{ALC}$. La interpretación $\mathcal{I}$ es un modelo en árbol *de C con respecto a $\mathcal{T}$ si $\mathcal{I}$ es un modelo de C con respecto a $\mathcal{T}$ y el grafo $\mathcal{G}_{\mathcal{I}}$ es un árbol cuya raíz pertenece a $C^{\mathcal{I}}$.*

Para $\mathcal{ALC}$ se cumple que todo concepto satisfacible en una TBox tiene un modelo en árbol. Para demostrarlo, debemos introducir la noción de *modelo desplegado* [1]. En el modelo desplegado los d-caminos constituirán los elementos del dominio.

Definición 4.12. *Sea $\mathcal{I}$ una interpretación y $d \in \Delta^{\mathcal{I}}$. El* modelo desplegado de $\mathcal{I}$ en d *es una interpretación $\mathcal{J}$ tal que:*

$$\Delta^{\mathcal{J}} = \{p : p \text{ es un } d\text{-camino en } \mathcal{I}\},$$
$$C^{\mathcal{J}} = \{p \in \Delta^{\mathcal{J}} : end(p) \in C^{\mathcal{I}}\} \text{ para todo } C \in \mathbf{C},$$
$$r^{\mathcal{J}} = \{(p, p') \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} : p' = (p, end(p')) \text{ y } (end(p), end(p')) \in r^{\mathcal{I}}\} \text{ para todo } r \in \mathbf{R}.$$

Para ilustrar este concepto usaremos un ejemplo. Sean $C, D \in \mathbf{C}$ y $r, s \in \mathbf{R}$; es fácil comprobar que la interpretación $\mathcal{I}$ expuesta en la parte izquierda de la Figura 4.1 es un modelo de C con respecto a la TBox

$$\mathcal{T} = \{C \sqsubseteq \exists r.D,\ D \sqsubseteq \exists r.C,\ C \sqcup D \sqsubseteq \exists s.\top\}$$

En la parte derecha de la Figura 4.1 tenemos a $\mathcal{J}$ el modelo desplegado de $\mathcal{I}$ en d. Como se puede intuir, se basa en considerar todos los d-caminos como nuevos elementos.

Es posible establecer una bisimulación entre una interpretación y su modelo desplegado.

[1] Del inglés *Unravelling model*.

Lema 4.13. *Sean $\mathcal{I}$ una interpretación, $d \in \Delta^{\mathcal{I}}$ y $\mathcal{J}$ el modelo deplegado de $\mathcal{I}$ en d. Entonces:*

$$\rho = \{(p, end(p)) \mid p \in \Delta^{\mathcal{J}}\}$$

es una bisimulación entre $\mathcal{J}$ e $\mathcal{I}$.

Demostración. Por la definición de la extensión de los conceptos en la interpretación $\mathcal{J}$ tenemos que $p \in A^{\mathcal{J}}$ si y sólo si $end(p) \in A^{\mathcal{I}}$ y por lo tanto la condición 1) de la definición de bisimulación se cumple.

Para comprobar que se cumple la condición 2) de la definición de bisimulación supongamos que $(p, p') \in r^{\mathcal{J}}$ y que $(p, e) \in \rho$. Ya que $end(p)$ es el único elemento de $\Delta^{\mathcal{I}}$ que está ρ-relacionado con p, necesariamente $e = end(p)$. Debemos ver entonces que existe $f \in \Delta^{\mathcal{I}}$ tal que $(p', f) \in \rho$ y $(end(p), f) \in r^{\mathcal{I}}$. Basta tomar $f = end(p')$, ya que $(p', end(p')) \in \rho$ por definición de ρ y por la definición de la extensión de los roles en la interpretación $\mathcal{J}$ tenemos que $(end(p), end(p')) \in r^{\mathcal{I}}$.

Para comprobar la condición 3) de la definición de bisimulación, supongamos que $(e, f) \in r^{\mathcal{I}}$ y $(p, e) \in \rho$ (luego $end(p) = e$). Debemos encontrar un d-camino p' tal que $(p', f) \in \rho$ y $(p, p') \in r^{\mathcal{J}}$. Definimos entonces la concatenación $p' = (p, f)$. Esto es un d-camino ya que p es un d-camino con $end(p) = e$ y $(e, f) \in r^{\mathcal{I}}$. Tenemos entonces que $end(p') = f$, lo que implica por definición de ρ que $(p', f) \in \rho$ y, por otro lado, $p' = (p, end(p'))$ y $(end(p), end(p')) \in r^{\mathcal{I}}$ ya que $end(p) = e$ y $end(p') = f$. Luego $(p, p') \in r^{\mathcal{J}}$. □

Proposición 4.14. *Dados $\mathcal{I}$ una interpretación, $d \in \Delta^{\mathcal{I}}$ y $\mathcal{J}$ el modelo desplegado de $\mathcal{I}$ en d, para todo concepto $\mathcal{ALC}$ C y todo d-camino $p \in \Delta^{\mathcal{J}}$, tenemos que*

$$p \in C^{\mathcal{J}} \text{ si y solo si } end(p) \in C^{\mathcal{I}}$$

Demostración. Es una consecuencia inmediata del Lema 4.13 y del Teorema 4.8. □

Teorema 4.15. *$\mathcal{ALC}$ cumple la propiedad de modelo en árbol, es decir, si $\mathcal{T}$ es una TBox $\mathcal{ALC}$ y C un concepto $\mathcal{ALC}$ tal que C es satisfacible con respecto a $\mathcal{T}$, entonces C tiene un modelo en árbol con respecto a $\mathcal{T}$.*

Demostración. Sean $\mathcal{I}$ un modelo de $\mathcal{T}$ y $d \in \Delta^{\mathcal{I}}$ tal que $d \in C^{\mathcal{I}}$ (nótese que ha de existir $d \in C^{\mathcal{I}}$ por ser C satisfacible respecto a $\mathcal{T}$). Probaremos que el modelo desplegado $\mathcal{J}$ de $\mathcal{I}$ en d es un modelo en árbol de C con respecto a $\mathcal{T}$.

Veamos primero que $\mathcal{J}$ es un modelo de $\mathcal{T}$. Consideremos una GCI $D \sqsubseteq E$ en $\mathcal{T}$ cualquiera, y veamos que si $p \in \Delta^{\mathcal{J}}$ satisface $p \in D^{\mathcal{J}}$, entonces $p \in E^{\mathcal{J}}$. Por la Proposición 4.14, puesto que $p \in \Delta^{\mathcal{J}}$ tenemos $end(p) \in D^{\mathcal{I}}$, lo que implica por la GCI que $end(p) \in E^{\mathcal{I}}$, ya que $\mathcal{I}$ es un modelo de $\mathcal{T}$. De nuevo por la Proposición 4.14 tenemos que $p \in E^{\mathcal{J}}$.

Veamos ahora que el grafo $\mathcal{G}_{\mathcal{J}}$ es un árbol con raíz d, donde d se interpreta como un d-camino de longitud 1. Primero, nótese que d es el único d-camino de longitud 1. En efecto, por la definición de las extensiones de roles en $\mathcal{J}$ y la definición de d-camino, todos y cada uno de los d-caminos de longitud > 1 tienen un predecesor con respecto a algún rol. En consecuencia, d es el único elemento sin predecesor, es decir, la raíz. Ahora, si tomamos p un d-camino de longitud > 1, por definición de d-camino $p = (p', end(p))$ donde p' es un d-camino de longitud ≥ 1 tal que existe un rol $r \in \mathbf{R}$ con $(end(p'), end(p)) \in r^{\mathcal{I}}$ y,

además, p' es único. Por lo tanto, p' es el único d-camino con $(p', p) \in \bigcup_{r \in \mathbf{R}} r^{\mathcal{J}}$, lo que completa la prueba de que $\mathcal{G}_{\mathcal{J}}$ es un árbol con raíz d.

Falta comprobar que la raíz d del árbol pertenece a la extensión de C en $\mathcal{J}$. Sin embargo, esto se sigue inmediatamente de la Proposición 4.14 ya que $d = end(d)$ y $d \in C^{\mathcal{I}}$. □

Nótese que bajo estas condiciones nada impide que el modelo en árbol sea infinito. Por ejemplo el concepto A con respecto a la TBox $\{A \sqsubseteq \exists r.A\}$ no acepta un modelo de árbol finito. En general, si el modelo inicial $\mathcal{I}$ tiene un ciclo, el árbol que surja del desplegado tendrá una rama infinita como se puede observar en el ejemplo de la Figura 4.1.

Cada concepto $\mathcal{ALC}$ satisfacible tiene un modelo en árbol. Esto puede generalizarse a una base de conocimiento: basta observar que acepta un modelo con varios árboles o *modelo en bosque* en el que se combinan modelos en árbol de distintos conceptos. Esta es la base del algoritmo de tableaux semánticos con el que buscaremos un modelo de una base de conocimiento con forma de bosque que cumpla ciertas propiedades.

Capítulo 5

Razonamiento con tableaux en $\mathcal{ALC}$

Como hemos comentado en el Capítulo 3, la lógica $\mathcal{ALC}$ es decidible. Además, hemos probado que todos los problemas de razonamiento planteados en la Definición 3.6 pueden transformarse en un problema de consistencia de una ontología o base de conocimiento. De entre la variedad de técnicas existentes para decidir sobre consistencia, el enfoque basado en tableaux es quizás el más conocido y hoy en día versiones optimizadas de este método se encuentran implementadas en diversos razonadores.

A continuación, tras describir los principios generales del enfoque de tableaux, presentaremos un algoritmo para la consistencia en $\mathcal{ALC}$ y probaremos que es correcto, terminando en un número finito de pasos, y que es completo.

5.1. Nociones básicas de tableaux.

Recordemos que una base de conocimiento $\mathcal{K}$ es consistente si existe algún modelo de $\mathcal{K}$. Por ejemplo, consideremos la base de conocimiento $\mathcal{K}_1$ en la Figura 5.2 y la interpretación $\mathcal{I}_1$ expuesta a continuación y representada en la Figura 5.1.

$$
\begin{aligned}
\Delta^{\mathcal{I}_1} &= \{jo, a, enPat, ca, \\
&\quad toPat, pat\}, \\
jose^{\mathcal{I}_1} &= jo, \\
alberto^{\mathcal{I}_1} &= a, \\
tortilla1^{\mathcal{I}_1} &= toPat, \\
\textit{la-ensalada}^{\mathcal{I}_1} &= enPat, \\
\mathsf{Verdura}^{\mathcal{I}_1} &= \{pat, enPat\}, \\
\mathsf{Carne}^{\mathcal{I}_1} &= \{ca\},
\end{aligned}
\qquad
\begin{aligned}
\mathsf{Tortilla}^{\mathcal{I}_1} &= \{toPat\}, \\
\mathsf{Persona}^{\mathcal{I}_1} &= \{jo, a\}, \\
\mathsf{Cocinero}^{\mathcal{I}_1} &= \{jo\}, \\
\mathsf{Comensal}^{\mathcal{I}_1} &= \{a\}, \\
\mathsf{Comida}^{\mathcal{I}_1} &= \{pat, ca, toPat, enPat\}, \\
\mathsf{Ensalada}^{\mathcal{I}_1} &= \{enPat\}, \\
cocina^{\mathcal{I}_1} &= \{(jo, toPat), (jo, enPat)\}, \\
contiene^{\mathcal{I}_1} &= \{(toPat, pat), (enPat, pat)\}.
\end{aligned}
$$

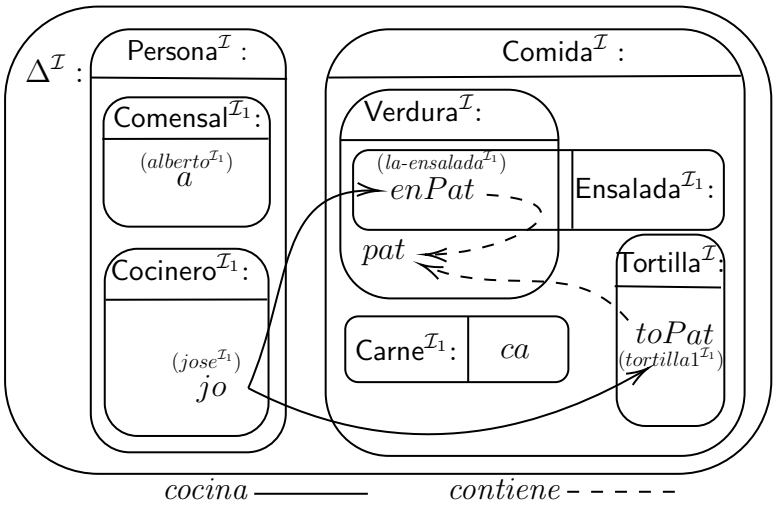

Figura 5.1: Diagrama de la interpretación $\mathcal{I}_1$.

$$
\begin{aligned}
\mathcal{T}_1 = \quad & \{\mathsf{Cocinero} \equiv \mathsf{Persona} \sqcap \exists cocina.\top, && (\mathcal{T}_1.1)\\
& \mathsf{Comida} \sqsubseteq \neg\mathsf{Persona}, && (\mathcal{T}_1.2)\\
& \mathsf{Comensal} \sqsubseteq \neg\mathsf{Cocinero}, && (\mathcal{T}_1.3)\\
& \exists contiene.\top \sqsubseteq \mathsf{Comida}\} && (\mathcal{T}_1.4)\\
\mathcal{A}_1 = \quad & \{jose : \mathsf{Persona} \sqcap \exists cocina.\mathsf{Verdura}, && (\mathcal{A}_1.1)\\
& jose : \forall cocina.(\mathsf{Verdura} \sqcup \neg\mathsf{Carne}), && (\mathcal{A}_1.2)\\
& tortilla1 : \neg\mathsf{Carne} \sqcap \neg\mathsf{Verdura}, && (\mathcal{A}_1.3)\\
& la\text{-}ensalada : \mathsf{Verdura} \sqcap \exists contiene.\mathsf{Verdura}, && (\mathcal{A}_1.4)\\
& (jose, tortilla1) : cocina, && (\mathcal{A}_1.5)\\
& (jose, la\text{-}ensalada) : cocina\} && (\mathcal{A}_1.6)
\end{aligned}
$$

Figura 5.2: Base de conocimiento $\mathcal{K}_1$

La existencia de $\mathcal{I}_1$ prueba que $\mathcal{K}_1$ es consistente, decimos que es un *testigo* de la consistencia de $\mathcal{K}_1$.

La idea básica de las técnicas de inferencia basadas en tableaux es probar la consistencia de una base de conocimiento demostrando la existencia de un testigo. Esto se hace constructivamente, comenzando con la ABox $\mathcal{A}$ y extendiéndola de forma que cumpla los axiomas de $\mathcal{T}$ manteniendo los de $\mathcal{A}$. Este procedimiento puede terminar con la construcción de un testigo que representa la ABox o con el descubrimiento de alguna contradicción que indique que ese testigo no existe (y por lo tanto, que $\mathcal{K}$ es inconsistente).

En las siguientes secciones se presentará un algoritmo que tiene como entrada una base de conocimiento $\mathcal{ALC}$, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ y que devuelve como salida si es consistente o no. Presentaremos primero un algoritmo para decidir la consistencia de la ABox y veremos cómo puede extenderse para decidir sobre la consistencia de la ontología completa.

En lo que sigue, supondremos que A, B, C, D son conceptos, r, $s \in \mathbf{R}$ nombres de

roles y a, b, c, $d \in \mathbf{I}$ nombres de individuos. Para facilitar el desarrollo y sin pérdida de generalidad se supondrá que los conceptos en $\mathcal{T}$ y $\mathcal{A}$ están en *forma normal negativa*, es decir, la negación sólo se aplicará a conceptos simples. Se puede reescribir cualquier concepto compuesto en esta forma usando las propiedades vistas en el Lema2.3.

5.2. Consistencia de una ABox $\mathcal{ALC}$

Supondremos que la ABox $\mathcal{A}$ cumple que: es no vacía y todo individuo concreto que aparece en $\mathcal{A}$ ocurre al menos en un aserto de la forma $a : C$. Una ABox que cumple estas condiciones se denomina ABox *normalizada*. Nótese que, aunque en la versión original de una ABox no vacía no aparezca un aserto para cada individuo, siempre podemos agregar asertos del tipo $a : \top$ hasta que quede normalizada. Igualmente, si es una ABox vacía podemos añadir un aserto $a : \top$ para un nuevo nombre de individuo.

Consideramos ahora el caso $\mathcal{K} = (\emptyset, \mathcal{A})$. La idea del algoritmo es ir añadiendo asertos en función de la semántica de los conceptos de la ABox inicial, lo que se denomina explicar la semántica. Para ello, expande los conceptos compuestos atendiendo a su estructura sintáctica, es decir, añade asertos para sus subconceptos según unas reglas de expansión hasta que no quede ninguno sin expandir o encontremos una contradicción. Para identificar estas situaciones definimos las siguientes nociones.

Definición 5.1. *Una ABox $\mathcal{A}$ contiene un* conflicto *si, para algún nombre de individuo a y para algún concepto C, $\{a : C, a : \neg C\} \subseteq \mathcal{A}$. Diremos que $\mathcal{A}$* está libre de conflictos *si no contiene ningún conflicto. $\mathcal{A}$ será* completa *si contiene un conflicto o si ninguna de las reglas de expansión de la Figura 5.3 es aplicable.*

La definición de ABox completa hace referencia a las reglas de expansión de la Figura 5.3, que están en consonancia con los constructores utilizados en la definición de sintaxis de la LD $\mathcal{ALC}$.

5.2.1. Algoritmo de tableaux semánticos

Con las reglas de expansión podemos definir un algoritmo para decidir la consistencia de una ABox $\mathcal{A}$ normalizada. Un primer enfoque no determinista nos daría una sencilla definición del algoritmo: *$\mathcal{A}$ es consistente si y solo si existe una forma de aplicar las reglas de expansión que nos lleven a una ABox completa y libre de conflicto.* Este algoritmo, sin embargo, no se puede implementar directamente. La razón es que confundiría situaciones de no determinismo relevantes en las que la elección puede afectar a la salida del algoritmo, con situaciones no relevantes, que no afectan al resultado. Por este motivo el algoritmo utilizará búsqueda para aplicar aquellas reglas que conlleven una elección no determinista.

En el caso de $\mathcal{ALC}$, el no determinismo se presenta en $\sqcup$-REGLA. El resto de reglas definen una única ampliación de la ABox, mientras que la $\sqcup$-REGLA define dos ampliaciones distintas que pueden llevar a una conclusión completamente diferente. Por ejemplo, basta considerar $\mathcal{A} = \{a : \neg C, a : C \sqcup D\}$; al aplicar $\sqcup$-REGLA, como resultado podemos obtener $\mathcal{A} = \{a : \neg C, a : C \sqcup D, a : C\}$ que tiene un conflicto, o $\mathcal{A} = \{a : \neg C, a : C \sqcup D, a : D\}$ que es completa y libre de conflictos. Por lo tanto, debemos explorar todas las ramas posibles en busca de un testigo en lugar de terminar en cuanto encontremos un conflicto. Así, el

$\sqcap$-REGLA (ABox $\mathcal{A}$):

$$\textbf{si} \begin{cases} 1. & a : C \sqcap D \in \mathcal{A} \text{ y} \\ 2. & \{a : C, a : D\} \nsubseteq \mathcal{A} \end{cases} \quad \textbf{entonces } \mathcal{A} \to \mathcal{A} \cup \{a : C, a : D\}$$

$\sqcup$-REGLA (ABox $\mathcal{A}$):

$$\textbf{si} \begin{cases} 1. & a : C \sqcup D \in \mathcal{A} \text{ y} \\ 2. & \{a : C, a : D\} \cap \mathcal{A} = \emptyset \end{cases} \quad \textbf{entonces} \quad \begin{array}{l} \mathcal{A} \to \mathcal{A} \cup \{a : X\} \\ \text{para algún } X \in \{C, D\}. \end{array}$$

$\exists$-REGLA (ABox $\mathcal{A}$):

$$\textbf{si} \begin{cases} 1. & a : \exists r.C \in \mathcal{A} \text{ y} \\ 2. & \nexists b \in \mathcal{A} \text{ tal que} \\ & \{(a, b) : r, b : C\} \subseteq \mathcal{A} \end{cases} \quad \textbf{entonces} \quad \begin{array}{l} \mathcal{A} \to \mathcal{A} \cup \{(a, d) : r, d : C\} \\ \text{donde } d \in \mathbf{I} \text{ se introduce como} \\ \text{nuevo nombre de elemento en } \mathcal{A}. \end{array}$$

$\forall$-REGLA (ABox $\mathcal{A}$):

$$\textbf{si} \begin{cases} 1. & \{a : \forall r.C, (a, b) : r\} \subseteq \mathcal{A} \text{ y} \\ 2. & b : C \notin \mathcal{A} \end{cases} \quad \textbf{entonces } \mathcal{A} \to \mathcal{A} \cup \{b : C\}$$

Figura 5.3: Reglas de expansión para consistencia de ABox.

algoritmo deberá utilizar backtracking para comprobar (recursivamente) la consistencia de las ABoxes que resulten de todas las posibles maneras de aplicar la $\sqcup$-REGLA, de modo que la ABox original será consistente si alguna de estas ABoxes apropiadas lo es. En cambio, el algoritmo no busca un orden de aplicación de las reglas, porque el orden no altera la consistencia de una ABox; simplemente buscamos la existencia de una cadena de reglas que nos lleven a una ABox completa y libre de conflictos. Sin embargo, este orden puede afectar gravemente a la eficiencia del algoritmo.

Para facilitar la descripción detallada del algoritmo, introducimos $exp(\mathcal{A}, \text{R}, \alpha)$, una función que toma como parámetros una ABox $\mathcal{A}$, una regla a aplicar R y un aserto α (o par de asertos) de $\mathcal{A}$ para los cuales aplicar la regla y, como resultado, la ampliación (o ampliaciones) de la ABox. Por ejemplo,

- $exp(\{a : \neg C, a : C \sqcup D\}, \sqcup\text{-REGLA}, a : C \sqcup D)$ devuelve un conjunto conteniendo las dos ABoxes: $\{\{a : \neg C, a : C \sqcup D, a : C\}, \{a : \neg C, a : C \sqcup D, a : D\}\}$.
- $exp(\{b : \neg D, a : \forall r.D, (a, b) : r\}, \forall\text{-REGLA}, \{\forall r.D, (a, b) : r\})$ devuelve un conjunto de un elemento: $\{\{b : \neg D, \forall r.D, (a, b) : r, b : D\}\}$.

Para reglas deterministas $exp()$ devuelve un conjunto de cardinal uno, mientras que para reglas indeterministas el cardinal es mayor que uno (en $\mathcal{ALC}$ sólo aparece cardinal dos, pero podría ser mayor en alguna extensión al definir las reglas de expansión para otros constructores sintácticos).

El algoritmo de consistencia se basará en buscar una expansión de la ABox inicial que sea completa y libre de conflictos usando esta función $exp()$. Informalmente, se trata de ir eligiendo reglas y aplicarlas a la ABox de forma recursiva, de modo que si se encuentra un

conflicto se vuelve hasta la última regla no determinista y se explora el resto de opciones. Como ya se ha dicho, la forma de elegir las reglas a la hora de aplicarlas puede beneficiar o empeorar la eficiencia del algoritmo. Por ello, a menudo se opta por usar algún tipo de heurística, como dar prioridad a las reglas deterministas. En nuestro caso expondremos el algoritmo en su forma más general.

Definición 5.2. *Definimos como* algoritmo de consistencia de ABox *al algoritmo expuesto en* Algoritmo 1 *que toma como input una ABox $\mathcal{A}$ normalizada y usa el algoritmo* expandir() *para aplicar las reglas de expansión (expuestas en* Figura 5.3*) a $\mathcal{A}$.*

Algoritmo 1 Consistencia de ABox

```
función consistente(ABox A normalizada):
    si expandir(A) ≠ ∅ entonces
        devolver "consistente"
    si no
        devolver "inconsistente"

función expandir(ABox A normalizada):
    si A no es completa entonces
        seleccionar una regla R que sea aplicable a A y un aserto
        o par de asertos α en A a los cuales R es aplicable
        si existe ABox A' ∈ exp(A, R, α) con expandir(A') ≠ ∅ entonces
            devolver expandir(A')
        si no
            devolver ∅
    si no
        si A contiene un conflicto entonces
            devolver ∅
        si no
            devolver A
```

Ilustramos ahora este algoritmo aplicándolo a la ABox $\mathcal{A}_1$ del ejemplo de la Figura 5.2, sin tener en cuenta la interpretación $\mathcal{I}_1$ ni la TBox $\mathcal{T}_1$. Nótese que la ABox ya está normalizada, por lo que podemos aplicar el algoritmo directamente.

Podemos comenzar aplicando, por ejemplo, $\sqcap$-REGLA en el primer, tercer y cuarto aserto. Obteniendo respectivamente: $\mathcal{A}_2 = \mathcal{A}_1 \cup \{jose : \mathsf{Persona}, jose : \exists cocina.\mathsf{Verdura}\}$; $\mathcal{A}_3 = \mathcal{A}_2 \cup \{tortilla1 : \neg\mathsf{Carne}, tortilla1 : \neg\mathsf{Verdura}\}$ y $\mathcal{A}_4 = \mathcal{A}_3 \cup \{la\text{-}ensalada : \mathsf{Verdura}, la\text{-}ensalada : \exists contiene.\mathsf{Verdura}\}$. Como hemos mencionado antes, el expandir en un aserto u otro no importa ya que no tenemos en cuenta el orden de las reglas aplicadas. Consideremos en nuestra ABox $\mathcal{A}$ añadidas las tres expansiones.

A continuación podríamos intentar aplicar la $\exists$-REGLA a los asertos $jose : \exists cocina.\mathsf{Verdura}$ y $la\text{-}ensalada : \exists contiene.\mathsf{Verdura}$. En el segundo caso se cumplen las precondiciones de la regla e introducimos un nuevo elemento x en los asertos $(la\text{-}ensalada, x) : contiene$ y $x : \mathsf{Verdura}$. Por otro lado, el en el primer caso encontramos en la ABox $(jose, la\text{-}ensalada) : cocina$ y $la\text{-}ensalada : \mathsf{Verdura}$, por lo que no podemos aplicar la$\exists$-REGLA a este aserto.

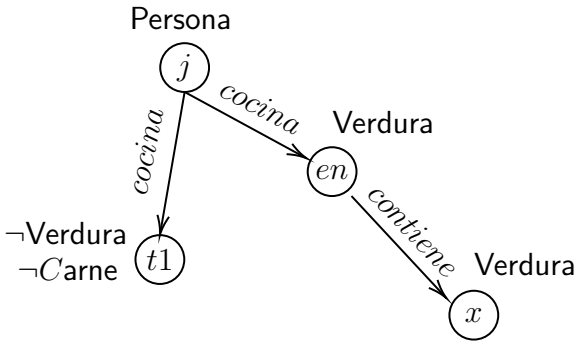

Figura 5.4: Diagrama del testigo generado por la ABox completa y libre de conflicto $\mathcal{A}_{ext}$ después de aplicar el algoritmo tableaux.

Cabe destacar que, si el orden de aplicación de las reglas hubiese sido distinto, podríamos haber generado un elemento nuevo que podría dar lugar a un testigo distinto.

En cuanto al segundo aserto $\mathcal{A}_1.2$ de la ABox inicial, aplicamos primero $\forall$-REGLA, añadiendo así a la ABox expandida los asertos: *la-ensalada* : $\mathsf{Verdura} \sqcup \neg\mathsf{Carne}$ y *tortilla*1 : $\mathsf{Verdura} \sqcup \neg\mathsf{Carne}$. Por ahora tenemos una ABox libre de conflictos, pero aún no está completa; debemos ver si se puede aplicar la regla indeterminista $\sqcup$-REGLA a cada uno de los asertos recién añadidos.

- En el caso de *la-ensalada* : $\mathsf{Verdura} \sqcup \neg\mathsf{Carne}$, en la ABox expandida ya figura el aserto *la-ensalada* : $\mathsf{Verdura}$. Por lo tanto, no debemos aplicar la regla.

- En el caso de *tortilla*1 : $\mathsf{Verdura} \sqcup \neg\mathsf{Carne}$ sucede lo mismo, ya que tenemos el aserto *tortilla*1 : $\neg\mathsf{Carne}$ y por lo tanto no es aplicable.

Al concluir estas expansiones hemos llegado a una ABox extendida, $\mathcal{A}_{ext}$, completa y libre de conflicto lo que significa que existe un testigo de nuestra ABox y $\mathcal{A}_1$ es satisfacible. Como ya sabíamos por la existencia de un modelo de la misma. El proceso completo se muestra en el diagrama de la Figura 5.5. Por otro lado, se expone el diagrama en árbol del testigo generado por nuestra ABox $\mathcal{A}_{ext}$ en la Figura 5.4.

5.2.2. Propiedades del algoritmo

Nótese que la ABox extendida tiene forma de bosque cuyos árboles poseen como raíz un nombre de individuo y cuyas aristas son los roles existentes. Al aplicar la $\exists$-REGLA introducimos nuevos nombres de individuos y ampliamos el árbol. Es por ello que nos referiremos a estos nuevos elementos como *individuos del árbol* y a los elementos que figuraban al inicio en la ABox *individuos raíz*. Definimos también de forma natural los términos *predecesor* y *sucesor* para referirnos a dos elementos relacionados por la $\exists$-REGLA como hemos visto en el ejemplo anterior. En dicho caso el elemento x sería el sucesor de a y, recíprocamente, el elemento a, predecesor de x. Usaremos *ancestro* y *descendiente* para hablar de la clausura transitiva de la relación de predecesor y sucesor.

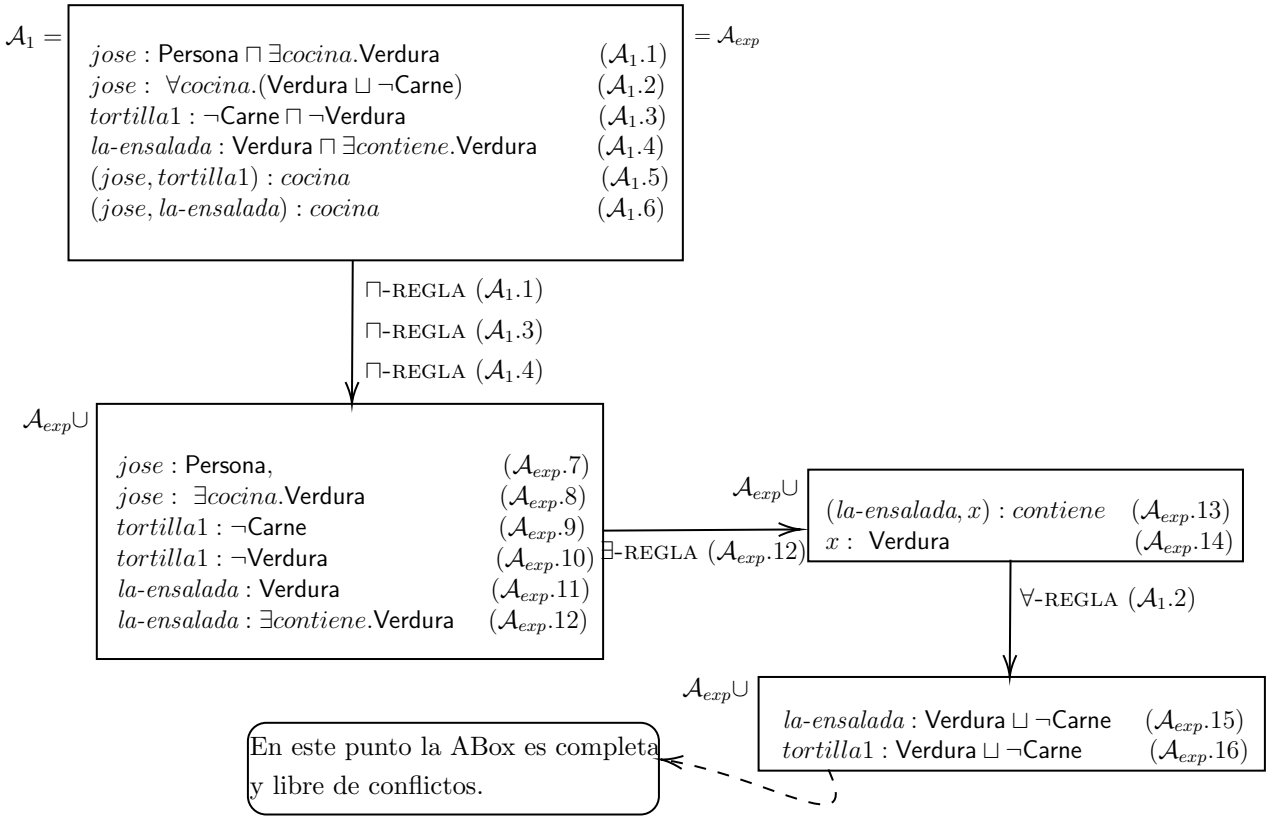

Figura 5.5: Representación del algoritmo aplicado al ejemplo de ABox $\mathcal{A}_1$.

Por último, denotaremos con $con_{\mathcal{A}}(a)$ al conjunto de conceptos C que aparecen en nuestra ABox $\mathcal{A}$ de la forma $a : C$, es decir,

$$con_{\mathcal{A}}(a) = \{C \mid a : C \in \mathcal{A}\}$$

El Lema 4.3 proporciona una cota para el número de subconceptos en una ABox.

$$|sub(\mathcal{A})| \leq \sum_{a:C \in \mathcal{A}} size(C)$$

Por lo tanto, el tamaño de $sub(\mathcal{A})$ depende linealmente del tamaño de $\mathcal{A}$.
Probaremos ahora que el algoritmo de consistencia es correcto, es completo y termina.

Lema 5.3. (Terminación) *Para cada ABox $\mathcal{ALC}$, $\mathcal{A}$, el algoritmo* consistente($\mathcal{A}$) *termina.*

Demostración. Sea $m = |sub(\mathcal{A})|$. La terminación del algoritmo es una consecuencia de las siguientes propiedades de las reglas de expansión:

1. Las reglas de expansión nunca eliminan un aserto de $\mathcal{A}$ y cada regla que aplicamos añade al menos un nuevo aserto de la forma $a : C$ para un individuo concreto a y para un concepto $C \in sub(\mathcal{A})$. Además, por el Lema 4.3 el tamaño de $sub(\mathcal{A})$ está acotado por el tamaño de $\mathcal{A}$, por lo tanto puede haber como mucho m aplicaciones de las reglas añadiendo un aserto de la forma $a : C$ para un individuo concreto a y por ello $|con_{\mathcal{A}}(a)| \leq m$.

2. Se añade un nuevo individuo a $\mathcal{A}$ sólo cuando se usa la $\exists$-REGLA en un aserto de la forma $a : C$ con C un concepto que contenga la restricción existencial. Como no puede haber más de m restricciones existenciales en $\mathcal{A}$, entonces un solo individuo puede provocar como mucho la adición de m nuevos individuos, por lo tanto el grado de expansión del árbol que forma nuestra ABox está acotado por m.

3. Las reglas $\exists$-REGLA y $\forall$-REGLA se aplican en asertos de la forma $a : \exists r.C$ o $a : \forall r.C$ y añaden asertos de la forma $b : C$ donde b es un sucesor de a. En ambos casos C es un subconcepto estricto de $\exists r.C$ o $\forall r.C$ y obviamente tenemos $size(C) < size(\exists r.C)$. Sucesivamente podremos obtener otros asertos del tipo $b : D$, pero de nuevo con D un subconcepto estricto de C. Por lo tanto, se cumple que siendo b un sucesor de a, tenemos $sub(con_{\mathcal{A}}(b)) \subseteq sub(con_{\mathcal{A}}(a))$. Además, el mayor concepto en $con_{\mathcal{A}}(a)$ tiene mayor tamaño que el mayor concepto en $con_{\mathcal{A}}(b)$, ya que b es un sucesor de a. Esta desigualdad de tamaño quiere decir que hay al menos un elemento en $sub(con_{\mathcal{A}}(a))$ que no está en $sub(con_{\mathcal{A}}(b))$, teniendo así el contenido estricto $sub(con_{\mathcal{A}}(b)) \subsetneq sub(con_{\mathcal{A}}(a))$. Como consecuencia la profundidad del árbol que generamos con nuestra ABox está acotada por m.

De acuerdo a estas propiedades, hay una cota dependiente del tamaño de $\mathcal{A}$ para los nombres de individuos que aparecen en la ABox (el número de árboles de nuestro modelo de bosque), el grado de expansión del árbol y su profundidad. Todo ello en conjunto nos asegura que hay una cota en el tamaño del bosque que surge de forma natural al usar nuestro algoritmo en una ABox. Es decir, tiene una cantidad finita de árboles y ramas que explorar y, por lo tanto, termina. □

Lema 5.4. (Corrección) *Si* consistente($\mathcal{A}$) *retorna "consistente", entonces* $\mathcal{A}$ *es consistente.*

Demostración. Sea $\mathcal{A}'$ el conjunto retornado por expandir($\mathcal{A}$). Dado que el algoritmo ha retornado "consistente", $\mathcal{A}'$ es una ABox completa y libre de conflicto. Veamos que existe una estrecha correspondencia entre $\mathcal{A}'$ y una interpretación $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ que es un modelo de $\mathcal{A}'$. Dado que las reglas de expansión no eliminan asertos, tenemos que $\mathcal{A} \subseteq \mathcal{A}'$, luego si $\mathcal{I}$ es un modelo de $\mathcal{A}'$ lo será también de $\mathcal{A}$ y, por tanto, un testigo de su consistencia.

Construiremos una interpretación a partir de $\mathcal{A}'$ como sigue:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \{a \mid a : C \in \mathcal{A}'\},\\
a^{\mathcal{I}} &= a \text{ para cada nombre de individuo } a \text{ que aparece en } \mathcal{A}',\\
A^{\mathcal{I}} &= \{a \mid A \in con_{\mathcal{A}'}(a)\} \text{ para cada concepto } A \in sub(\mathcal{A}'),\\
r^{\mathcal{I}} &= \{(a,b) \mid (a,b) : r \in \mathcal{A}'\} \text{ para cada rol } r \text{ que aparece en } \mathcal{A}'.
\end{aligned}$$

Nótese que cada nombre de individuo a que aparece en $\mathcal{A}'$ lo hace en un aserto del tipo $a : C$; los individuos raíz lo hacen por la definición de ABox normalizada y los elementos del árbol por cómo está definida la $\exists$-REGLA.

Es fácil ver que $\mathcal{I}$ es una interpretación. Puesto que la ABox está normalizada, no está vacía y habrá al menos un aserto de la forma $a : C$, luego $\Delta^{\mathcal{I}} \neq \emptyset$. Por construcción $\cdot^{\mathcal{I}}$ lleva todo individuo concreto en $\mathcal{A}'$ a un elemento de $\Delta^{\mathcal{I}}$, todo concepto $A \in sub(\mathcal{A}')$ a un subconjunto de $\Delta^{\mathcal{I}}$ y todo rol r a un subconjunto de $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

$\mathcal{I}$ será un modelo de $\mathcal{A}'$ si satisface cada aserto en $\mathcal{A}'$. Por construcción, todos los asertos de roles se cumplen. Para los asertos de concepto procedemos por inducción en la estructura de los conceptos para demostrar que si $a : C \in \mathcal{A}'$, entonces $a^{\mathcal{I}} \in C^{\mathcal{I}}$.

Si C es un concepto simple, por definición de $\mathcal{I}$ tenemos que $C^{\mathcal{I}} = \{a \mid C \in con_{\mathcal{A}'}(a)\}$ y $a \in \{a \mid C \in con_{\mathcal{A}'}(a)\}$ ya que tenemos el aserto $a : C$, por lo tanto $a^{\mathcal{I}} = a \in C^{\mathcal{I}}$.

Si C es un concepto compuesto, supongamos por hipótesis de inducción que se cumple para todos sus subconceptos propios. Pueden darse los siguientes casos:

- $C = \neg D$: ya que $\mathcal{A}'$ es libre de conflictos, $a : \neg D \in \mathcal{A}'$ implica que $a : D \notin \mathcal{A}'$. Debido a que todos los conceptos están en forma normal negativa, D es un concepto simple. Por definición de $\mathcal{I}$, tenemos $a^{\mathcal{I}} \notin D^{\mathcal{I}}$, lo que implica que $a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}} = C^{\mathcal{I}}$.

- $C = D \sqcup E$: si $a : D \sqcup E \in \mathcal{A}'$, puesto que $\mathcal{A}'$ es completa, $\{a : D, a : E\} \cap \mathcal{A}' \neq \emptyset$ (de otra forma la $\sqcup$-REGLA sería aplicable). Entonces $a^{\mathcal{I}} \in D^{\mathcal{I}}$ o $a^{\mathcal{I}} \in E^{\mathcal{I}}$ por inducción y por lo tanto, $a^{\mathcal{I}} \in D^{\mathcal{I}} \cup E^{\mathcal{I}} = (D \sqcup E)^{\mathcal{I}}$.

- $C = D \sqcap E$: análogo al caso anterior.

- $C = \forall r.D$: sea $a : \forall r.D \in \mathcal{A}'$ y consideremos b con $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$; tenemos que demostrar que $b^{\mathcal{I}} \in D^{\mathcal{I}}$. Por definición de $\mathcal{I}$ tenemos que $(a,b) : r \in \mathcal{A}'$, por lo tanto $b : D \in \mathcal{A}'$ (de otra forma podríamos aplicar la $\forall$-REGLA) y por inducción se cumple que $b^{\mathcal{I}} \in D^{\mathcal{I}}$. Como esto es aplicable a todo b tal que $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, $a^{\mathcal{I}} \in (\forall r.D)^{\mathcal{I}}$ por la semántica de $\forall$.

- $C = \exists r.D$: si $a : \exists r.D \in \mathcal{A}'$, puesto que la ABox es completa, debe existir algún b tal que $\{(a,b) : r, b : D\} \subset \mathcal{A}'$ (si no la $\exists$-REGLA sería aplicable). El aserto $(a,b) : r$

implica directamente que $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, y por inducción, el aserto $b : D$ implica que $b^{\mathcal{I}} \in D^{\mathcal{I}}$. Por lo tanto, por la semántica de $\exists$, $a^{\mathcal{I}} \in (\exists r.D)^{\mathcal{I}}$.

En consecuencia $\mathcal{I}$ satisface todos los asertos en $\mathcal{A}'$ y por lo tanto en $\mathcal{A}$, con lo que es un modelo de $\mathcal{A}$, lo que asegura la consistencia de la ABox. □

Lema 5.5. (Completitud) *Si $\mathcal{A}$ es consistente, entonces* consistente($\mathcal{A}$) *retorna "consistente".*

Demostración. Si $\mathcal{A}$ es consistente, existe un modelo $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ de $\mathcal{A}$. Además, como $\mathcal{A}$ es consistente, no puede contener un conflicto.

Si $\mathcal{A}$ es completa, expandir($\mathcal{A}$) simplemente retorna $\mathcal{A}$ y consistente($\mathcal{A}$) retorna "consistente". Si $\mathcal{A}$ no es completa, entonces se comenzarán a aplicar las reglas de expansión hasta que $\mathcal{A}$ sea completa. Veamos ahora que cada regla conserva la consistencia de la ABox:

- $\sqcup$-REGLA: Si $a : C \sqcup D \in \mathcal{A}$, entonces $a^{\mathcal{I}} \in (C \sqcup D)^{\mathcal{I}}$ luego $a^{\mathcal{I}} \in C^{\mathcal{I}}$ o $a^{\mathcal{I}} \in D^{\mathcal{I}}$. Por lo tanto, al menos una de las ABoxes $\mathcal{A}' \in exp(\mathcal{A}, \sqcup\text{-REGLA}, a : C \sqcup D)$ es consistente y por lo tanto al menos una de las llamadas de expandir() es aplicada a una ABox consistente.

- $\sqcap$-REGLA: Si $a : C \sqcap D$, entonces $a^{\mathcal{I}} \in (C \sqcap D)^{\mathcal{I}}$ luego $a^{\mathcal{I}} \in C^{\mathcal{I}}$ y $a^{\mathcal{I}} \in D^{\mathcal{I}}$. Por lo tanto $\mathcal{I}$ sigue siendo un modelo de $\mathcal{A} \cup \{a : C, a : D\}$, que es la ABox que retorna expandir(). En consecuencia, $\mathcal{A}$ sigue siendo consistente después de aplicar la regla.

- $\exists$-REGLA: Si $a : \exists r.C \in \mathcal{A}$, entonces $a^{\mathcal{I}} \in (\exists r.C)^{\mathcal{I}}$ luego existe algún $x \in \Delta^{\mathcal{I}}$ tal que $(a^{\mathcal{I}}, x) \in r^{\mathcal{I}}$ y $x \in C^{\mathcal{I}}$. Por lo tanto, existe un modelo $\mathcal{I}'$ de $\mathcal{A}$ que extiende $\mathcal{I}$ de forma que para algún nuevo nombre de individuo d $d^{\mathcal{I}'} = x$. Este modelo lo es también de $\mathcal{A} \cup \{(a, d) : r, d : C\}$, con lo que $\mathcal{A}$ sigue siendo consistente después de aplicar la regla.

- $\forall$-REGLA: Si $\{a : \forall r.C, (a, b) : r\} \subseteq \mathcal{A}$, entonces $a^{\mathcal{I}} \in (\forall r.C)^{\mathcal{I}}$ y $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, luego $b^{\mathcal{I}} \in C^{\mathcal{I}}$. Por lo tanto, $\mathcal{I}$ sigue siendo un modelo de $\mathcal{A} \cup \{b : C\}$ y $\mathcal{A}$ sigue siendo consistente después de aplicar la regla.

□

Teorema 5.6. *El algoritmo de tableaux en Algoritmo 1 es un procedimiento de decisión para la consistencia de ABoxes $\mathcal{ALC}$.*

Demostración. Que el algoritmo es un procedimiento de decisión para ABoxes $\mathcal{ALC}$ normalizadas se sigue directamente de los lemas precedentes y, como hemos visto al principio de la sección, toda ABox $\mathcal{ALC}$ se puede transformar en una ABox equivalente normalizada. □

La complejidad del algoritmo expuesto es exponencial en tiempo y en espacio, pero puede adaptarse para usar espacio polinomial, usando lo que se llama *trace technique* [9]. Se basa en usar primero todas las reglas $\sqcap, \sqcup, \forall$-REGLA, ya que el orden de las reglas no altera el resultado, y finalmente explorar las ramas generadas con $\exists$-REGLA reusando el espacio para explorar cada rama, ya que pueden considerarse problemas independientes.

5.3. Consistencia de una ontología $\mathcal{ALC}$

Extendemos ahora el algoritmo de tableaux para decidir sobre la consistencia de una base de conocimiento $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ cualquiera.

Nótese que como consecuencia del Lema 3.8, tenemos que

$$\mathcal{I} \text{ satisface } C \sqsubseteq D \text{ si y solo si } \mathcal{I} \text{ satisface } \top \sqsubseteq D \sqcup \neg C,$$
$$\mathcal{I} \text{ satisface } C \equiv D \text{ si y solo si } \mathcal{I} \text{ satisface } \top \sqsubseteq (D \sqcup \neg C) \sqcap (C \sqcup \neg D).$$

Por lo tanto, podemos asumir sin pérdida de generalidad que los axiomas de nuestra TBox serán de la forma $\top \sqsubseteq E$. Extendemos entonces nuestra noción de *normalizada* a TBoxes y bases de conocimiento. Diremos que una TBox $\mathcal{T}$ está *normalizada* si sus axiomas son de la forma $\top \sqsubseteq E$, donde E está en NNF; y que una base de conocimiento $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ está *normalizada* si ambas $\mathcal{T}$ y $\mathcal{A}$ están normalizadas.

5.3.1. Algoritmo de tableaux semánticos

Necesitamos ampliar las reglas de expansión para tratar los axiomas de la TBox. Añadimos una nueva regla, la $\sqsubseteq$-REGLA e introducimos una pequeña modificación en la $\exists$-REGLA para asegurar que el algoritmo termina. El conjunto resultante de reglas de expansión se encuentra en la Figura 5.6. Para poder comprenderlo, necesitamos realizar algunas reflexiones sobre cómo afecta la nueva regla $\sqsubseteq$-REGLA a la ABox expandida y definir la noción de bloqueo.

La modificación de la $\exists$-REGLA viene dada por los ciclos que pueden surgir de la combinación de las reglas $\sqsubseteq$-REGLA y $\exists$-REGLA en su definición original de la Figura 5.3. En efecto, podría darse el caso de que haya un sucesor b de a tal que el conjunto de subconceptos $sub(con_{\mathcal{A}}(b))$ no sea un subconjunto estricto de $sub(con_{\mathcal{A}}(a))$, y por lo tanto la profundidad del árbol correspondiente en la ABox podría no estar acotada. Por ejemplo, consideremos la base de conocimiento $(\{A \sqsubseteq \exists r.A\}, \{a : A\})$ normalizada como $(\{\top \sqsubseteq \neg A \sqcup \exists r.A\}, \{a : A\})$. Si aplicamos consecutivamente las reglas $\sqsubseteq$-REGLA y $\exists$-REGLA en su versión inicial, generaremos un elemento b que es indistinguible de a respecto a nuestra base de conocimiento, por lo tanto podemos volver a aplicar las reglas a b repitiendo este proceso indefinidamente.

Si una situación como la anterior aparece y la ABox sigue estando libre de conflictos, podemos parar este ciclo y usar la regularidad de las ramas generadas para definir una ABox (y respectivamente un modelo) sin introducir clones de a. Para formalizar esta idea y asegurar la terminación introducimos la técnica conocida como *bloqueo*.

Definición 5.7. (Bloqueo $\mathcal{ALC}$) *Un nombre de individuo b en una ABox $\mathcal{A}$ está* bloqueado por un nombre de individuo *a si a es un ancestro de b y $con_{\mathcal{A}}(b) \subseteq con_{\mathcal{A}}(a)$.*

Un nombre de individuo b está bloqueado en *$\mathcal{A}$ si está bloqueado por algún nombre de individuo a o si alguno de sus ancestros está bloqueado en $\mathcal{A}$.*

Tenemos dos consecuencias inmediatas de esta definición: cuando un individuo está bloqueado, todos sus descendientes también lo están; y, como un individuo raíz no tiene ancestros, nunca podrá estar bloqueado. El bloqueo garantiza la terminación del algoritmo sin comprometer su corrección. Así, construiremos un modelo a partir de una ABox

$\sqcap$-REGLA (KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$):

$$\textbf{si} \left\{ \begin{array}{ll} 1. & a : C \sqcap D \in \mathcal{A} \text{ y} \\ 2. & \{a : C, a : D\} \nsubseteq \mathcal{A} \end{array} \right\} \quad \textbf{entonces } \mathcal{A} \to \mathcal{A} \cup \{a : C, a : D\}$$

$\sqcup$-REGLA (KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$):

$$\textbf{si} \left\{ \begin{array}{ll} 1. & a : C \sqcup D \in \mathcal{A} \text{ y} \\ 2. & \{a : C, a : D\} \cap \mathcal{A} = \emptyset \end{array} \right\} \quad \textbf{entonces} \quad \begin{array}{l} \mathcal{A} \to \mathcal{A} \cup \{a : X\} \\ \text{para algún } X \in \{C, D\} \end{array}$$

$\exists$-REGLA (KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$):

$$\textbf{si} \left\{ \begin{array}{ll} 1. & a : \exists r.C \in \mathcal{A} \text{ y} \\ 2. & \nexists b \in \mathcal{A} \text{ tal que} \\ & \{(a, b) : r, b : C\} \nsubseteq \mathcal{A} \\ 3. & a \text{ no está bloqueado} \end{array} \right\} \quad \textbf{entonces} \quad \begin{array}{l} \mathcal{A} \to \mathcal{A} \cup \{(a, d) : r, d : C\} \\ \text{donde } d \in \mathbf{I} \text{ se introduce} \\ \text{como nuevo elemento en } \mathcal{A}. \end{array}$$

$\forall$-REGLA (KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$):

$$\textbf{si} \left\{ \begin{array}{ll} 1. & \{a : \forall r.C, (a, b) : r\} \subseteq \mathcal{A} \text{ y} \\ 2. & b : C \notin \mathcal{A} \end{array} \right\} \quad \textbf{entonces } \mathcal{A} \to \mathcal{A} \cup \{b : C\}$$

$\sqsubseteq$-REGLA (KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$):

$$\textbf{si} \left\{ \begin{array}{ll} 1. & a : C \in \mathcal{A}, \top \sqsubseteq D \in \mathcal{T} \text{ y} \\ 2. & a : D \notin \mathcal{A} \end{array} \right\} \quad \textbf{entonces } \mathcal{A} \to \mathcal{A} \cup \{a : D\}$$

Figura 5.6: Reglas de expansión para consistencia de una base de conocimiento normalizada cualquiera.

contemplando el caso de un bloqueo. Si b está bloqueado por a tenemos dos opciones posibles:

- repetir la estructura de la sección comprendida entre a y b infinitamente y por lo tanto tener un modelo de bosque infinito, o
- en lugar de introducir b, la rama forma un ciclo de vuelta a a, llevándonos a un modelo finito con ciclos.

La modificación de la $\exists$-REGLA en la Figura 5.6 opta por la segunda opción, manteniendo un modelo finito a costa de sacrificar la estructura en árbol.

Basta entonces con modificar el algoritmo expuesto en Algoritmo 1 de consistencia de ABoxes simplemente sustituyendo las reglas de expansión originales por las de la Figura 5.6. El resultado se muestra en el Algoritmo 2.

Definición 5.8. *Definimos como* algoritmo de consistencia de bases de conocimiento $\mathcal{ALC}$ *al algoritmo expuesto en* Algoritmo 2 *que toma como entrada una base de conocimiento* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *normalizada y usa el algoritmo* expandir() *para aplicar las reglas de expansión de la Figura 5.6 a* $\mathcal{K}$.

Algoritmo 2 Consistencia de base de conocimiento $\mathcal{ALC}$

```
función consistente(KB K = (T, A) normalizada):
    si expand(T, A) ≠ ∅ entonces
        devolver "consistente"
    si no
        devolver "inconsistente"

función expandir(KB K = (T, A) normalizada):
    si A no es completa entonces
        seleccionar una regla R que sea aplicable a A y un aserto
        o par de asertos α en A a los cuales R es aplicable
        si existe ABox A' ∈ exp(A, R, α) con expand(T, A') ≠ ∅ entonces
            devolver expandir(T, A')
        si no
            devolver ∅
    si no
        si A contiene un conflicto entonces
            devolver ∅
        si no
            devolver A
```

5.3.2. Propiedades del algoritmo

Se probará ahora, para cualquier base de conocimiento $\mathcal{ALC}$, que el algoritmo termina, es correcto y es completo.

Lema 5.9. (Terminación) *Para cada base de conocimiento $\mathcal{ALC}$ $\mathcal{K}$,* consistente($\mathcal{K}$) *termina.*

Demostración. La prueba es muy similar a la prueba del Lema 5.3, variando únicamente la cota superior de la profundidad alcanzada por el árbol. Sea $m = |sub(\mathcal{K})|$. Por cómo se definen las reglas de expansión en la Figura 5.6, se cumple que $con_{\mathcal{A}}(a) \subseteq sub(\mathcal{K})$ y, por lo tanto, a lo sumo hay 2^m subconjuntos diferentes de $con_{\mathcal{A}}(a)$. Se tienen tres propiedades:

1. Puede haber como mucho m aplicaciones de las reglas de expansión a cualquier individuo concreto dado (véase Lema 5.3).

2. El grado de ramificación de cada árbol está acotado por m (véase Lema 5.3).

3. Puesto que $con_{\mathcal{A}}(a) \subseteq sub(\mathcal{K})$ y $|sub(\mathcal{K})| = m$, todo camino entre individuos del árbol en la ABox expandida puede contener como mucho 2^m individuos (véase la cota del número máximo de individuos diferentes en la prueba del Lema 5.3) antes de que contenga dos individuos a y b tales que b sea un descendiente de a y $con_{\mathcal{A}}(b) \subseteq con_{\mathcal{A}}(a)$, es decir, que b y todos sus descendientes estén bloqueados. Por lo tanto la profundidad de cada árbol está acotada por 2^m.

La segunda y la tercera propiedad implican que el número de individuos que puede generarse es finito y la primera, que las reglas a aplicar sobre ellos también son finitas. Todo ello implica que el algoritmo termina. □

Lema 5.10. (Corrección) *Si* consistente($\mathcal{K}$) *retorna "consistente", entonces $\mathcal{K}$ es consistente.*

Demostración. Como en la prueba del Lema 5.4, usaremos la ABox $\mathcal{A}'$ completa y libre de conflictos retornada por expandir($\mathcal{K}$) para construir un modelo $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ de $\mathcal{K}$, con la única dificultad añadida de considerar los nombres de individuos bloqueados. Podemos resumirlo en dos pasos: primero, definir una nueva ABox $\mathcal{A}''$ completa y libre de conflictos a partir de $\mathcal{A}'$ pero tratando los casos de individuos bloqueados añadiendo un aserto de rol que forme un ciclo; y segundo, construir un modelo a partir de $\mathcal{A}''$.

Definimos $\mathcal{A}''$ como sigue:

$$\begin{aligned}\mathcal{A}'' = &\{a : C \mid a : C \in \mathcal{A}' \text{ y } a \text{ no está bloqueado }\} \cup \\ &\{(a,b) : r \mid (a,b) : r \in \mathcal{A}' \text{ y } b \text{ no está bloqueado }\} \cup \\ &\{(a,b') : r \mid (a,b) : r \in \mathcal{A}' \ a \text{ no está bloqueado y } b \text{ está bloqueado por } b'\}.\end{aligned}$$

Es fácil ver que $\mathcal{A} \subseteq \mathcal{A}''$ ya que $\mathcal{A} \subseteq \mathcal{A}'$ y para todos los asertos $a : C$ y $(a,b) : r$ en $\mathcal{A}$, ambos elementos a y b son individuos raíz y, por lo tanto, no pueden estar bloqueados.

Nótese también que ningún nombre de individuo en $\mathcal{A}''$ está bloqueado: para los asertos $a : C$ se cumple por definición y para los asertos de roles $(a,b) : r$, distinguimos dos casos. Primero, si $(a,b) : r \in \mathcal{A}'$ y b no está bloqueado, entonces obviamente a no puede estar bloqueado (puesto que si lo estuviese, b también lo estaría al ser su sucesor). En el segundo caso, es decir, $(a,b) : r \in \mathcal{A}'$, a no está bloqueado y b está bloqueado por b' basta probar que b' no está bloqueado. Pero, puesto que b está bloqueado y a no lo está, necesariamente b está bloqueado por algún predecesor b' de b en $\mathcal{A}'$. Dado que $b' = a$ o b' es un predecesor de a, el hecho de que a no esté bloqueado implica que b' no puede estarlo.

La siguiente propiedad es una consecuencia inmediata de la construcción de $\mathcal{A}''$:

$$con_{\mathcal{A}''}(a) = con_{\mathcal{A}'}(a) \qquad (*)$$

Como $\mathcal{A}'$ es libre de conflictos, entonces $\mathcal{A}''$ también lo es, puesto que, por la propiedad $(*)$, si hubiera algún conflicto en $\mathcal{A}''$ también lo habría en $\mathcal{A}'$. Además, que $\mathcal{A}'$ sea completa implica que $\mathcal{A}''$ también lo es:

- Para la regla $\sqcap$-REGLA: si $a : C \sqcap D \in \mathcal{A}''$, entonces por la propiedad $(*)$ $a : C \sqcap D \in \mathcal{A}'$ y como $\mathcal{A}'$ es completa, $\{a : C, a : D\} \subseteq \mathcal{A}'$. De nuevo por la propiedad $(*)$, tenemos que $\{a : C, a : D\} \subseteq \mathcal{A}''$
- Para las reglas $\sqcup$-REGLA y $\sqsubseteq$-REGLA la prueba es análoga a la anterior.
- Para la $\exists$-REGLA: si $a : \exists r.C \in \mathcal{A}''$, entonces $a : \exists r.C \in \mathcal{A}'$ y a no está bloqueado en $\mathcal{A}'$. Como $\mathcal{A}'$ es completa, ha de existir b tal que $\{(a, b) : r, b : C\} \subseteq \mathcal{A}'$. Distinguimos ahora dos casos:
  - Si b no está bloqueado, entonces $\{(a, b) : r, b : C\} \subseteq \mathcal{A}''$.
  - Si b está bloqueado, como ya hemos visto, debe existir un individuo b' que no está bloqueado y es predecesor de b. Entonces $(a, b') : r \in \mathcal{A}''$ y $con_{\mathcal{A}'}(b) \subseteq con_{\mathcal{A}'}(b')$, lo que, junto con la propiedad $(*)$, implica que $b' : C \in \mathcal{A}''$, obteniendo $\{(a, b') : r, b' : C\} \subseteq \mathcal{A}''$.

  En ambos casos, no se puede aplicar la regla $\exists$-REGLA en $\mathcal{A}''$.
- Para la $\forall$-REGLA: Si $\{a : \forall r.C, (a, b') : r\} \subseteq \mathcal{A}''$, entonces $a : \forall r.C \in \mathcal{A}'$ y ninguno de los individuos a o b están bloqueados en $\mathcal{A}'$. Distinguimos dos casos:
  - Si $(a, b') : r \in \mathcal{A}'$, entonces $b' : C \in \mathcal{A}'$, ya que es completa y por la propiedad $(*)$, $b' : C \in \mathcal{A}''$.
  - Si $(a, b') : r \notin \mathcal{A}'$, entonces existe b tal que $(a, b) : r \in \mathcal{A}'$ con b bloqueado por b' en $\mathcal{A}'$, y puesto que la ABox $\mathcal{A}'$ es completa, $b : C \in \mathcal{A}'$. Entonces, por la definición de bloqueo, $b' : C \in \mathcal{A}'$ y por la propiedad $(*)$ tenemos $b' : C \in \mathcal{A}''$.

  En ambos casos la $\forall$-REGLA no es aplicable en $\mathcal{A}''$.

Construimos ahora el modelo $\mathcal{I}$ a partir de $\mathcal{A}''$, al igual que en el Lema 5.4:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \{a \text{ es un individuo concreto en } \mathcal{A}''\},\\
a^{\mathcal{I}} &= a \text{ para cada individuo concreto } a \text{ que aparece en } \mathcal{A}'',\\
A^{\mathcal{I}} &= \{a \mid A \in con_{\mathcal{A}''}(a)\} \text{ para cada concepto simple } A \in \mathcal{A}'',\\
r^{\mathcal{I}} &= \{(a, b) \mid (a, b) : r \in \mathcal{A}''\} \text{ para cada rol } r \text{ que aparece en } \mathcal{A}'.
\end{aligned}$$

$\mathcal{I}$ es un modelo de $\mathcal{K}$ si es un modelo de $\mathcal{T}$ y de $\mathcal{A}$. La prueba de que $\mathcal{I}$ así definido es un modelo de $\mathcal{A}''$ y, por lo tanto, de $\mathcal{A}$ es equivalente a la demostración en el Lema 5.4. Falta ver que $\mathcal{I}$ es un modelo de $\mathcal{T}$, es decir, que satisface cada GCI en $\mathcal{T}$. Para cada GCI $\top \sqsubseteq D \in \mathcal{T}$ y cada individuo concreto a en $\mathcal{A}''$, tenemos que $a : D \in \mathcal{A}''$ (de otra forma, la regla $\sqsubseteq$-REGLA sería aplicable) y, como tenemos que $\mathcal{I}$ es un modelo de nuestra ABox $\mathcal{A}''$, entonces $a = a^{\mathcal{I}} \in D^{\mathcal{I}}$. Puesto que a es un elemento arbitrario de $\Delta^{\mathcal{I}}$, tenemos que $\Delta^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ como buscábamos. □

Lema 5.11. (Completitud) *Si $\mathcal{K}$ es consistente, entonces* consistente($\mathcal{K}$) *retorna "consistente".*

Demostración. Análogamente a como razonamos en la prueba del Lema 5.5, las aplicaciones recursivas de expandir() conservan la consistencia. El bloqueo es irrelevante en este caso, solo influye en que la construcción termine. La única diferencia reside en la regla $\sqsubseteq$-REGLA: si $a : C \in \mathcal{A}$ y $\top \sqsubseteq D \in \mathcal{T}$, entonces por la semántica de $\mathcal{ALC}$ $a^{\mathcal{I}} \in D^{\mathcal{I}}$ en todo modelo $\mathcal{I}$ de $(\mathcal{T}, \mathcal{A})$, luego $\mathcal{I}$ sigue siendo un modelo de $(\mathcal{T}, \mathcal{A} \cup \{a : D\})$. □

Teorema 5.12. *El algoritmo presentado en* Algoritmo 2 *es un procedimiento de decisión sobre la consistencia de bases de conocimiento $\mathcal{ALC}$.*

Demostración. Se sigue directamente de los Lemas 5.9, 5.10 y 5.11. □

Con respecto a la complejidad de este algoritmo tenemos dos cuestiones cruciales. Primero, la transformación de la TBox en una TBox normalizada puede (como mucho) duplicar el tamaño de la base de conocimiento. Por otro lado, como hemos visto en la demostración del Lema 5.9, el algoritmo genera árboles cuyo grado de ramificación está acotado por m y cuya profundidad está acotada por 2^m, siendo m el número de subconceptos en nuestra base de conocimiento $\mathcal{K}$. En consecuencia, el algoritmo requiere espacio que es doblemente exponencial al tamaño de la entrada.

Existen optimizaciones del algoritmo que mejoran su eficiencia aunque dichas optimizaciones quedan fuera del ámbito de este trabajo; pueden consultarse más detalles en la parte final de la subsección 4.2.3 de [3].

Capítulo 6

Conclusiones

Este trabajo cubre los conceptos necesarios para iniciarse en el mundo de las lógicas descriptivas a través de la lógica $\mathcal{ALC}$, cumpliendo así con el objetivo inicial del trabajo.

Tras motivar su estudio por su relevancia en representación del conocimiento, hemos definido el lenguaje $\mathcal{ALC}$. Hemos visto cómo formar una descripción a partir de nombres de conceptos y nombres de roles mediante el uso de constructores (sintaxis) y cómo dotar a estas descripciones de significado a través de interpretaciones (semántica).

Una vez definida la sintaxis y semántica del lenguaje hemos expuesto cómo construir una base de conocimiento u ontología para describir un dominio. Hemos identificado la parte terminológica (TBox) y la parte asertiva (ABox) y discutido las propiedades de las mismas. A partir de una base de conocimiento hemos planteado una serie de problemas de razonamiento y hemos visto que todos se reducen al problema de consistencia de una ontología. Para comprender mejor la lógica $\mathcal{ALC}$, hemos visto que puede identificarse con un fragmento decidible de la lógica de predicados. También hemos comentado algunas limitaciones en la expresividad de $\mathcal{ALC}$ y hemos presentado las cinco extensiones más conocidas de este lenguaje.

A continuación, hemos introducido algunas nociones de teoría de modelos que nos han permitido estudiar algunas propiedades relevantes de las ontologías en $\mathcal{ALC}$, como pueden ser cotas a su tamaño o la existencia de un modelo en árbol.

Por último, hemos presentado el método de tableaux semánticos para decidir sobre el problema de consistencia de una ontología. Primero, hemos descrito un algoritmo para decidir sobre la consistencia de una ABox y hemos probado que siempre termina y es correcto y completo. A continuación hemos ampliado este algoritmo a cualquier ontología demostrando que se mantienen sus propiedades. Obtenemos así un método para decidir sobre la consistencia de una base de conocimiento y, por lo tanto, resolver diversos problemas de razonamiento sobre dicha base.

Las lógicas descriptivas constituyen un campo muy amplio y, necesariamente, son muchos los temas que han quedado fuera de este trabajo.

Por ejemplo, en el caso de las ampliaciones de $\mathcal{ALC}$, se ha expuesto su sintáxis y semántica pero no se han mencionado las consecuencias que tiene este aumento de la expresividad en la decidibilidad de la LD ampliada.Tampoco hemos abordado cómo se ampliaría un método tableaux para incluir los nuevos constructores. Faltaría también probar que esos constructores no son mero azúcar sintáctico: para probar las contenciones estrictas entre lógicas descriptivas y el aumento en expresividad al incorporar las distintas

ampliaciones se usa el concepto de bisimulación expuesto en el Capítulo 4.

Hay varios aspectos que quedan sin explorar en este trabajo debido a su extensión. Cabe destacar demostrar la decibilidad del fragmento FO^2 de la lógica de predicados con el que se identifica $\mathcal{ALC}$ así como demostrar la decibilidad de $\mathcal{ALC}$ directamente a partir de la cota del Lema 4.6. También podría estudiarse la clase de complejidad en la que se enmarcan los problemas de razonamiento de una LD en función de los constructores (la expresividad) que se incorporan al lenguaje. Estos resultados consistirían, en sí mismos, un tema suficientemente rico como para abordarlos por sí mismos en otros trabajos fin de grado. Además, también existen LDs más expresivas que $\mathcal{ALC}$ para los que todavía no se conocen, como puede comprobarse en el navegador de complejidad de `www.cs.man.ac.uk/~ezolin/dl`.

Se ha comentado también que el lenguaje de ontologías OWL fundamentado en LDs constituye un estándar utilizado en muchos campos. Hay líneas de investigación que buscan ofrecer una expresividad más reducida pero permitiendo establecer cotas más bajas para la complejidad de la inferencia, dando lugar así a las LDs-*lite*. En general, este trabajo aborda una pequeña parte de una amplia disciplina en la que todavía hay muchos problemas abiertos.

Para terminar, destacamos las diversas materias y conocimientos del Grado en Matemáticas relevantes para este trabajo. Entre ellos, cabe destacar: la teoría básica de conjuntos esencial en la semántica de conceptos; nociones sobre relaciones que surgen al tratar con roles y con el concepto de bisimulación; conocimiento sobre algoritmos y sus propiedades; planteamientos, técnicas y conceptos de representación del conocimiento dentro de los sistemas inteligentes; nociones sobre lenguajes formales; una base de pensamiento lógico transversal a todos los estudios del grado y conocimiento sobre grafos (árboles) y sus propiedades.

Bibliografía

[1] F. Baader, I. Horrocks y U. Sattler. "Description Logics". En: *Handbook of Knowledge Representation*. Ed. por F.van Harmelen, V.Lifschitz y B. Porter. Elsevier, 2008. Cap. 3, págs. 135-179.

[2] F. Baader y W. Nutt. "Basic Description Logics". En: *The Description Logic Handbook: Theory, Implementation and Applications*. Ed. por F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi y P. F. Patel-Schneider. New York, NY, USA: Cambridge University Press, 2003.

[3] F. Baader, I. Horrocks, C. Lutz y U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[4] A. Borgida. "On the relative expressiveness of description logics and predicate logics". En: *Artificial Intelligence* 82 (1996), págs. 353 -367.

[5] S. Cramerotti, M. Turrini, M. Buccio, S. Larentis, M. Rospocher, L. Serafini, E. Cardillo e I. Donadello. "ePlanning: an Ontology-based System for Building Individualized Education Plans for Students with Special Educational Needs". En: *MED - Media Education* (2015).

[6] S. Derriere, A. Richard y A. Preite-Martinez. "An ontology of astronomical object types for the Virtual Observatory". En: *Highlights of Astronomy* 14 (2007), págs. 603-603.

[7] E. Grädel, P. G. Kolaitis y M. Y. Vardi. "On the Decision Problem for Two-Variable First-Order Logic". En: *The Bulletin of Symbolic Logic* 3 (1997), págs. 53-66.

[8] M. Krötzsch, F. Simancik e I. Horrocks. "Description Logics". En: *IEEE Intelligent Systems* 29 (2014), págs. 12-19.

[9] M. Schmidt-Schauß y G. Smolka. "Attributive concept descriptions with complements". En: *Artificial Intelligence* 48 (1991), págs. 13-17.

[10] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer y S. Katz. "Reengineering Thesauri for New Applications: the AGROVOC Example". En: *Journal of Digital Information* 4 (2006).